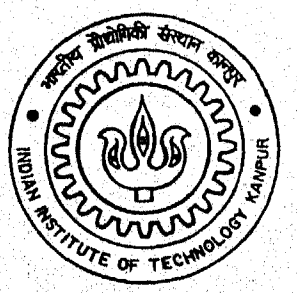
ASSESSMENT OF A PRECONDITIONED CONJUGATE GRADIENT ALGORITHM FOR MATRIX INVERSION AND ITS APPLICATION TO SOLUTAL TRANSPORT IN A CRYSTAL GROWTH PROCESS

By

Vinay Kumar



NUCLEAR ENGINEERING AND TECHNOLOGY PROGRAMME

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

AUGUST, 2003

CERTIFICATE

It is certified that the work contained in the thesis entitled "**Assessment of a Preconditioned Conjugate Gradient Algorithm for Matrix Inversion and its Application to Solutal Transport in a Crystal Growth Process**", by Vinay Kumar, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

K. Muralidhar
Professor
Dept. of Mechanical Engineering
I.I.T. Kanpur
Kanpur 208016

V. D. Puranik
Head
Environmental Assessment Division
Bhabha Atomic Research Center
Mumbai 400085

August, 2003

To Sumit and Vijay ...

ACKNOWLEDGMENT

With immense pleasure and great respect, I express my deep sense of gratitude to Professor K. Muralidhar, the teacher with profound intellect and diverse interests whose brilliant guidance throughout the period of my M. Tech. thesis helped me to surpass all the obstacles smoothly. He introduced me to the subject of this thesis, guided and assisted me in difficult times and taught me the basics of research. His company not only enriched my knowledge but also widened my ways of thinking.

I would like to express my sincere gratitude, regards and thanks to Shri V. D. Puranik and Shri A. K. Mahendra for many invaluable suggestions, constant encouragement and generous help at all stages of my research work.

I acknowledge my indebtedness to Mr. Jyotirmay Banerjee for his help with modeling transport phenomena in Czochralski crystal growth. I also acknowledge Dr. Tanuja Sheorey for enhancing my understanding of conjugate gradients method.

I am also thankful to Mr. Malay Kumar Das for his generous help and discussions throughout my research work.

I would like to thank Mr. Arvind Rao, Mr. Rajneesh Bhardwaj, Mr. Andalib Tariq, Mr. Sushanta Dutta, Mr. Atul Srivastava and Mr. Amar Singh for their pleasant company in the Laboratory.

I am thankful to all my friends and NET classmates for making my stay at IIT Kanpur enjoyable.

I must use this opportunity to acknowledge my indebtedness to my friends Sumit and Vijay who constantly motivated me with their series of e-mails.

Vinay Kumar

Abstract

In almost all numerical simulations matrix inversion is the most recursive and computationally intensive step. Therefore, the choice of matrix inverter, or equivalently the linear system solver is a critical step towards reducing the computational time of the overall simulation. Conjugate Gradient method is an extremely effective technique for solving linear systems when the coefficient matrix is symmetric and positive definite. Generalization and efficient implementation of conjugate gradient method for the linear systems with unsymmetric coefficient matrices is still an active area of research. One such generalization, based on preconditioning with incomplete lower-upper (ILU) decomposition of the coefficient matrix, has been proposed by Kershaw [1978]. Following this generalization, Sheorey [2001] has developed a Preconditioned Conjugate Gradient (PCG) algorithm that is believed to work well for the case of any nonsingular coefficient matrix.

The aim of the present research is to assess the above mentioned PCG algorithm for matrix inversion in various test cases including the complex problem of Czochralski crystal growth simulation. Czochralski technique is quite popular for growing single crystals that are of great importance for modern electronic and electro-optical applications and devices. Performance of these devices is deleteriously influenced by the presence of solute inhomogeneities in the crystals used. In order to investigate the solute segregation in these crystals, the coupled problem of heat transfer, hydrodynamics and solute conservation must be simultaneously solved. This research presents a macroscopic model for solute segregation.

CPU time analysis for the test case of unsteady-nonlinear heat conduction

shows that PCG is extremely effective for matrix inversion as compared to Gaussian elimination and Gauss-Seidel iterations. PCG becomes an even more attractive choice on finer grids. Further PCG is found to be well suited for matrix inversion in advection-diffusion problems. The results obtained for the fully developed Nusselt number in channel flow matches closely with the analytical solution reported in the literature. A comparison of the eigenvalue spectra of the original and preconditioned matrices show that preconditioning with incomplete lower-upper (ILU) decomposition is very effective in lowering the condition number towards unity. Moreover, this preconditioning technique is general in the sense that it can be constructed and applied to any nonsingular coefficient matrix.

For growth of a Nd:YAG crystal from its melt, the PCG algorithm has been used to determine the quasi-steady flow and temperature fields. In addition, the distribution of Nd particles in YAG has been determined by a truly unsteady calculation. Results obtained in this part of the study show the following:

In the simulations of Czochralski crystal growth, the pull velocity is varied with time to grow constant diameter crystals. The results obtained for pull-velocity variation show that it is not possible to maintain favorable growth conditions over a longer growth period. This is because the pull velocity continuously decreases with time. After a small increase in crystal length, melting starts. Therefore, to grow longer crystals with constant diameter, other parameters such as the crucible temperature and the enclosure temperature need to be varied, while keeping pull velocity positive and a constant. Redistribution of the segregated solute is found to correlate with the complex flow patterns in the melt. Profiles of the radial solute distribution reveal that crystal rotation improves radial uniformity of solute concentration in the grown crystals, while the uniformity along the crystal length is unaffected.

Contents

List of Figures

Nomenclature

A	Coefficient matrix	
b	Right hand side vector	
Bi_r	Radiation Biot number	$\epsilon_{eff}\sigma_r({T_{surf}}^2+{T_\infty}^2)(T_{surf}+T_\infty)b/k$
C_p	Specific heat, J/Kg.K	
Gr	Grashof number	$g\beta b^3(T_h-T_f)/{\nu_0}^2$
H	Melt-crystal interface height	H^*/b
Pr	Prandtl number	ν_0/α_0
Re	Reynolds number	$\omega b^2/\nu_0$
Rr	Radial ratio	R/b
S	Source term	
Sc	Schmidt number	ν_0/D_0
Ste_l	Liquid Stefan number	$C_{pl}(T_h-T_f)/h_{sl}$
Ste_s	Solid Stefan number	$C_{ps}(T_\infty-T_f)/h_{sl}$
T	Temperature, K	
Γ,	Diffusion coefficient	
a	Crystal radius, m	
b	Crucible radius, m	
h_{sl}	Latent heat of fusion, J/Kg.K	
g	Acceleration due to gravity, m^2/s	
k	Thermal conductivity, W/mK	
k_0	Segregation Coefficient	
L	Lower triangular matrix	
p	Pressure	$P^*b^2/\rho_0{\nu_0}^2$
r	Radial distance	r^*/b
t	Time	$t^*\nu_0/b^2$
u	Velocity in axial direction	u^*b/ν_0
U	Upper triangular matrix	
v	Velocity in radial direction	v^*b/ν_0
w	Velocity in swirl direction	w^*b/ν_0
z	Axial distance	z^*/b
$\mathbf{e}_z$	Unit vector in z-direction	
$\mathbf{e}_r$	Unit vector in r-direction	
$\mathbf{n}$	Unit normal vector	
α	Thermal diffusivity, m^2/s	

β	Thermal expansion coefficient, 1/K	
ρ	Density, Kg/m^3	
ϕ	General variable	
ϵ	Emissivity	
ν	Kinematic viscosity, m^2/s	
σ	Stefan-Boltzman constant, $W/(m^2K^4)$	
θ	Dimensionless temperature	$(T - T_f)/(T_h - T_f)$

Special symbols

$\|\ldots\|$	Absolute value
$\langle\ldots\|\ldots\rangle$	Inner product
$\|\|\ldots\|\|$	Norm

Subscripts

P	Node P
c	Crucible
eff	Effective
f	Freezing temperature
g	Gas
l	Melt
o	reference
s	Crystal

Superscripts

k	Iteration counter
$*$	Dimensionless

Chapter 1

Introduction

In all numerical simulations, the result of the discretization process is a system of algebraic equations, which can be linear or non-linear according to the nature of the partial differential equation(s) that govern the physical process of interest. In the non-linear case, the discretized equations must be solved by an iterative technique that involve guessing a solution, linearizing the equations about that solution, and improving the solution until a converged result is obtained. Moreover, if the mathematical problem is unsteady, these nonlinearity iterations need to be carried out at every time step. Thus in numerical simulations, matrix inversion is the most recursive and computationally intensive step and therefore, the choice of matrix inverter, or equivalently the linear system solver is a critical step towards reducing the computational time of the overall simulation.

Matrix inverters can be classified as direct and iterative. Direct methods are usually a variation of the Gaussian elimination technique, making use of forward elimination and backward substitution. It is equivalent to the decomposition of the coefficient matrix into lower and upper triangular factors. Direct inverters involve larger number of arithmetic operations per inversion and hence carry large round-off errors. Moreover, direct solvers when applied to a sparse system, introduce a large number of additional nonzero entries into the coefficient matrix. These "*fill-in*" values destroy the sparse structure of the coefficient matrix and thus increases the storage requirements significantly. However direct methods obtain the solution in finite number of predetermined steps and are generally stable. On the other hand, the term "*iterative methods*" refers to a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system at each step. The computational effort per inversion of an

iterative technique depends on its convergence rate. The rate of convergence of these methods depends greatly on the eigenvalue spectrum of the coefficient matrix. Hence, iterative methods usually involve a second matrix that transforms the coefficient matrix into one with a more favorable spectrum. This technique is called "*preconditioning*" and the transformation matrix is called a "*preconditioner*". A good preconditioner improves the convergence rate of an iterative method, sufficiently to overcome the extra cost of constructing and applying the preconditioner. When using an iterative method, the coefficient matrix is involved (directly or indirectly) only in terms of matrix by vector products. Thus there is no need for storing the zero elements of the coefficient matrix, and therefore, it is possible to implement the algorithms that are extremely cost-effective with respect to computing time as well as storage. In addition of the matrix by vector product, many iterative solvers are based on the vector operations like the calculation of a inner product. Thus except for some possible problems related to the global communication needed for the computation of inner products, such iterative methods are well suited for parallel and vectorized implementations. But then iterative methods are only conditionally stable. To combine the benefits of both the direct and iterative approaches, hybrid methods have been developed. The "*Preconditioned Conjugate Gradient (PCG)*" is one such technique that has been assessed in the present study.

1.1 Literature Review on Matrix Inversion

Lanczos [1952] and Hestenes and Stiefel [1952] discovered the method of Conjugate Gradients (CG) for solving linear systems with symmetric and positive definite coefficient matrix A. This method was initially regarded as an exact (direct) solution method, guaranteed to converge in no more than n steps, where n is the dimension of the problem. More precisely, in exact arithmetic the method is guaranteed to compute the solution to $Ax = b$ in exactly m steps, where m is the degree of minimal polynomial of A, *i.e.*, the number of distinct eigenvalues of A (since A is assumed to be symmetric). Based on numerical experiments it was observed that in presence of rounding errors, a few more than n iterations were often necessary to obtain an accurate solution, particularly on ill-conditioned systems. On the other hand, Hestenes and Stiefel [1952] observed that on well

conditioned systems, satisfactory approximations could be obtained in far fewer than n iterations, making the Conjugate Gradient method an attractive alternative to Gaussian Elimination. In spite of this, for a number of reasons the Conjugate Gradient method saw relatively little use for almost 20 years, during which Gaussian Elimination (for dense matrices) and SOR and Chebyschev iteration (for sparse matrices) were usually the preferred solution methods. Great details of these developments can be found in the interesting paper on the history of conjugate gradient method by Golub and O'Leary [1989].

This situation finally changed around 1970 with the publication of a paper of Reid. Reid [1971] shown that for large sparse matrices that are reasonably well conditioned, the Conjugate Gradient method is a very powerful scheme that yields good approximate solutions in far fewer than n steps. Reid's paper set the stage for the rehabilitation of the Conjugate Gradient method and stimulated research in various directions. On one hand, extensions to linear systems with indefinite and/or nonsymmetric matrices were sought; on the other, techniques to improve the conditioning of linear systems were developed in order to improve the rate of convergence of the method.

Early ideas of preconditioning the Conjugate Gradient method can be found in Engeli *et al.* [1959], and occasionally in several papers published during 1960s (Golub and O'Leary [1989]). A detailed study of SSOR preconditioning as a means of accelerating the convergence was carried out in by Axelsson [1972].

A measure breakthrough took place around the mid-1970s, with the introduction by Meijerink and van der Vorst of the incomplete Cholesky-Conjugate Gradient (ICCG) algorithm. Incomplete factorization methods were introduced for the first time by Buleev in the then-Soviet Union in the late 1950s, and independently by Varga in 1970s. However, Meijerink and van der Vorst [1977] deserve credit for recognizing the potential of incomplete factorizations as preconditioners for the Conjugate Gradient method. Another paper that did much to popularize this approach is by Kershaw [1978]. Since then, a number of improvements and extensions have been made, including level-of-fills and drop tolerances-based incomplete factorizations, generalizations to block matrices, modified and stabilized variants and even (more recently) efficient parallel implementations.

Then in late 1970s, an extensive research was under way for generalized CG methods that could be applied to nonsymmetric and/or indefinite problems. By

the early 1980s there was a new plethora of new versions of generalized conjugate gradient methods. However, none of these methods for general nonsymmetric problems worked in practice as well as the CG method did for symmetric positive definite problems. For some problems either a very large number of iterations were required or the method did not even converge. In his work Kershaw also proposed a generalization of the Conjugate Gradient algorithm for any non-singular coefficient matrix arising from partial differential equations. Sheorey T. [2001] followed this generalization and developed an incomplete *LU* preconditioned version of the conjugate gradient method that is applicable for the case of any non-singular coefficient matrix. The same version of PCG has been assessed in the present study. Some applications of this PCG algorithm can be found in Sheorey *et al.* [2001], Sheorey and Muralidhar [2001, 2003].

Books by Axelsson [1994], Barrett *et al.* [1993], Saad [1996], Golub *et al.* [1996] and Watkins [2002] provide excellent references for iterative matrix inversion in general and PCG in particular. Pissanetzky [1984] provides detailed discussions of almost all popular storage formats for sparse matrix handling.

1.2 Application Problem: Czochralski Crystal Growth

The growth of semiconductor single crystals is the basis for device fabrication in microelectronics industry. Single crystals of oxides have also become important materials for high power laser applications. These oxide crystals are found to have special optical, fluorescent, electrical and mechanical properties, and therefore, these crystals are finding new applications in a vast field of ferroelctric, pyroelectric, piezoelectric , acoustic, magnetic and opto-magnetic devices. Most of these applications require a high level of microstructural and chemical perfection that only can be exhibited by single crystals.

Although several methods have been devised for growing single crystals, the Czochralski (Cz) method has virtually dominated the entire production of single crystals. The popularity of this method comes from its ability to meet the stringent requirements for purity, doping, electrical and mechanical properties, and crystallographic perfection.

The basic principle of growing a single crystal by the Czochralski method

is shown in Figure 1.1. A crucible is initially filled with the material and then thermal energy is supplied by a heater surrounding the crucible. This raises the temperature of the solid above its melting temperature. A seed crystal, at the end of the rod, is then dipped into the melt and, after appropriate start-up procedures, slowly withdrawn from the melt. To keep the feeding material molten, the crucible is maintained at a constant temperature above the melting point.

Under suitable temperature conditions re-crystallization in the form of a single crystal occurs. This method bears Czochralski's name, who in 1916 established it as a means to determine the crystallization velocity of metals.

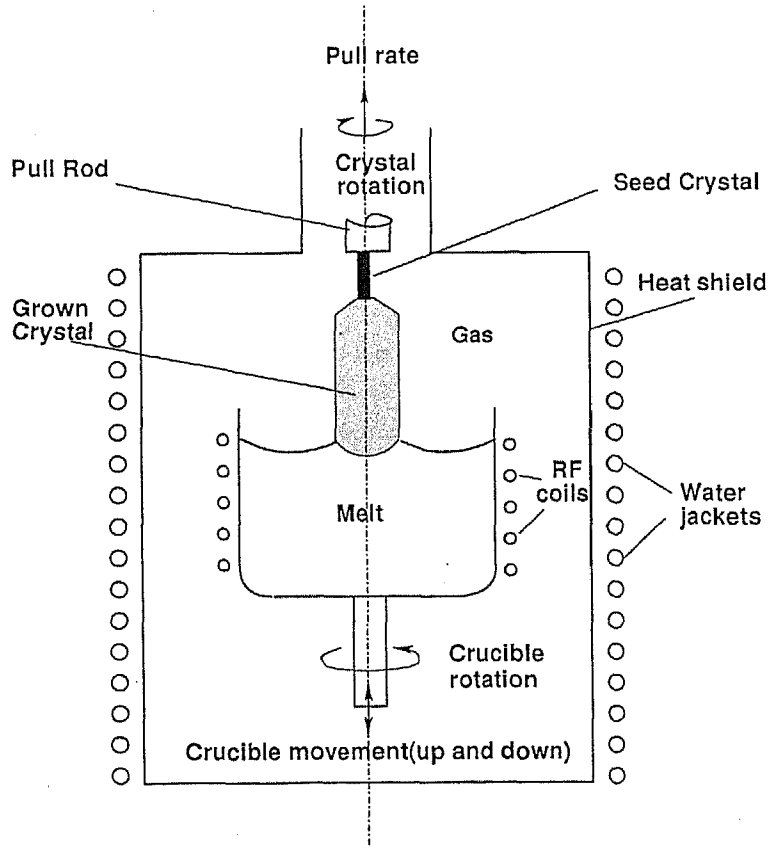


Figure 1.1: A schematic of the basic principle of Czochralski crystal growth process.

Convection in Czochralski melt is pervasive in all terrestrial growth systems. Sources for flow include buoyancy-driven convection caused by the solute and temperature dependence of the density; surface tension gradients along fluid-

melt menisci; forced convection introduced by the motion of solid surfaces, such as crucible and crystal rotation in the Cz systems; and motion of the melt induced by the solidification of the material. These flows are important causes of convection of heat and species and can have a dominant influence on the temperature field in the system and on solute incorporation into the crystal. A schematic of various transport mechanisms of Czochralski process is shown in Figure 1.2.

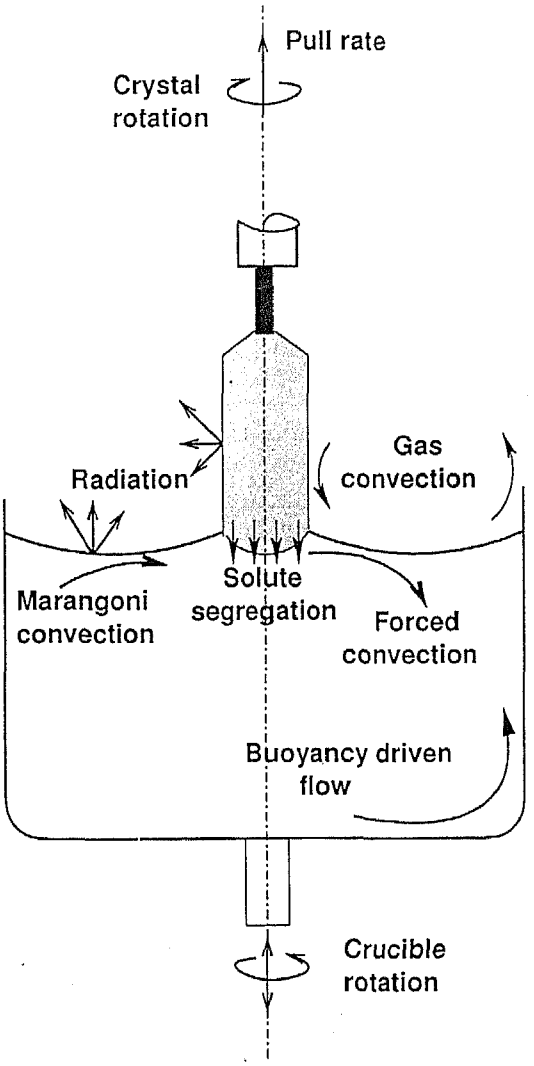


Figure 1.2: Schematic of various transport mechanisms in a Czochralski system.

A thorough understanding of heat and mass transport is a prerequisite to the optimization of Czochralski crystal growth systems for control of crystal quality, as measured by the degree of crystallographic perfection of the lattice and the spatially uniformity of electrically active solutes in it. Fluid motion in Cz-system

is of concern to crystal growers as it determines the transport of solute , heat, impurities, etc. to the grown crystal. Magnitude of stresses in the grown crystal is governed by the temperature gradients in the melt. It is currently accepted that the transport properties of the melt are the primary factors affecting the quality of grown crystals. These properties of melt are themselves determined by a broad and interacting matrix of variables that include crystal rotation, crucible rotation, pull rate, crucible temperature, pressure, orientation etc. Thus, state-of-the-art crystal growth requires algorithms that models the growth by incorporating the effects of all parameters and gives a set of values of these parameters as best possible guidelines for growing a particular crystal.

1.3 Literature Review on Crystal Growth

Over the past two decades, numerical simulations of Czochralski (Cz) crystal growth have been performed with varying degrees of complexity. Both finite difference and finite element methods have been used to simulate first the two dimensional and then the three-dimensional melt flows with many different kind of boundary conditions. A regular cylindrical geometry has usually been considered in most of the cases. Recently, efforts have been devoted to account for the crucible bottom curvature. Finite element methods can easily handle the non-flatness of the crucible bottom. However, this task becomes difficult if a regular finite difference or finite volume method is used. Consideration of crucible shape is important because the melt in industrial Cz(Si) process can become quite shallow. Also as the crystal is pulled, the aspect ratio of the melt continuously decreases.

Other difficulties in modeling realistic Cz flows arise from the fact that the positions of the melt-crystal interface and melt meniscus are neither known *a priori* nor planar. The melt-crystal interface coincides with the freezing point isotherm in the system (for a pure material), and the melt meniscus results from a force balance between the surface tension, pressure force and viscous effects. To complicate this further, the flow and temperature fields which are used to determine the shapes and locations of interface and meniscus are often oscillatory in nature. To overcome the difficulties associated with the time varying domains and irregular boundaries, different techniques have been employed by various

investigators.

Derby and Brown [1986] have developed an "integrated hydrodynamic thermocapillary" (IHTCM) model based on the finite element method which allows the calculation of interface as well as meniscus shape. The IHTCM models heat transfer by conduction and diffuse, gray radiation between all components within the Cz puller and axisymmetric, steady-state, laminar flow in the melt. The radius of the crystal is computed simultaneously and self-consistently with the heat transfer throughout the system and convection in the melt. Kinney and Brown [1993] have incorporated k-ϵ model for turbulent flow into the IHTCM. More recently, a global conduction-radiation procedure based on a finite element program ABAQUS, and a thermal radiation model, combined with a three-dimensional Navier-Stokes solver based on the finite element method has been developed by Koai *et al.* [1994].

In the area of finite differences Kopetsch [1989, 1990] extended a numerical procedure to include the deformable shapes of the melt-crystal interface and melt-ambient meniscus. His numerical scheme is based on an adaptive grid generation that maps the irregular boundary/interface movement and generates a coordinate transformation for a curvilinear domain. Several papers have been published in this direction and many of the problems faced in the formulation have been resolved. For example, the original scheme is suitable only for a single domain and steady-state problems. It is however, very difficult to use this formulation for a multizone system since it needs an adaptive grid generation algorithm in the interface zone. Also, the algorithm is not suitable for transient processes since it lacks the relationship between the grids at different times.

Zhang *et al.* [1995] developed a multizone adaptive grid-generation (MAGG) technique for efficient simulations of various transport phenomena on irregularly shaped domains and in regions with moving and/or free boundaries. MAGG is based on a variational optimization approach and provides boundary-fitting capability and discretionary control over grid distribution, as well a local and global orthogonality. Based on this grid generation technique and curvilinear finite volume discretization, Prasad *et al.* [1995] developed a high resolution computer model (MASTRAPP2d) to simulate crystal growth processes at low and high pressures. This scheme allows consideration of a multiphase system for more than one material in one single domain which may consist of irregular and

moving boundaries, interfaces and free surfaces.

Solute segregation in the Cz systems is coupled with melt convection, species diffusion and the physical and chemical properties of impurities in melt and solid. An accurate simulation of segregation remains a difficult task. The major challenge in predicting the solute distribution in the crystal stems from the fact that solutal transport in melt is an intrinsically dynamic process which is strongly time dependent. Thus dopant distribution cannot be treated as quasi-static like flow and temperature fields, which do not change much in a small time interval since the crystal rates are generally very small.

The equilibrium segregation coefficient k_0 is applicable only for negligibly slow growth rates. For finite or higher solidification rates, solute atoms with $k_0 < 1$ are rejected by the advancing solid at a greater rate than they can diffuse into the melt. Many attempts therefore have been made to obtain appropriate values of the effective segregation coefficient. For 1D (longitudinal) steady-state diffusion of a solute, Hurle [1993] gives

$$k_{eff} = \frac{k_0}{[1 - (1 - k_0)\, J]} \tag{1.1}$$

where

$$J = \left[\frac{V_p(\nu/\omega_s)^{1/2}}{D}\right] \int \exp\left[Sc - \int_0^x H(z)dz\right] dz \tag{1.2}$$

and ω_s is the rotation rate for the crystal and H is a rotation parameter.

Burton *et al.* [1953] assumed that solute distribution varies within a thin layer of thickness δ adjacent to solid-melt interface. A complete mixing caused by convection is assumed outside this boundary. They related the parameter J to the boundary layer thickness δ by

$$\frac{V_p\delta}{D} = -\ln(1 - J) \tag{1.3}$$

This approach was criticized by Wilson [1980], who has demonstrated that the relationship should be

$$\frac{V_p \delta}{D} = J \tag{1.4}$$

These two definitions (Equation 1.3 and Equation 1.4) approach each other as J becomes very small compared with unity. An evaluation of the integral in Equation 1.2 then yields

$$\delta = 1.6 Sc^{1/3} \left(\frac{\nu}{\omega}\right)^{1/2} \tag{1.5}$$

This gives the famous BPS model as

$$k_{eff} = \frac{k_0}{[k_0 + (1 + k_0)\exp(-V_p \delta / D)]} \tag{1.6}$$

where it has been assumed that $\rho_s = \rho_l$.

In the limit when $\delta \rightarrow 0$, k_{eff} becomes equal to k_0; that is, the boundary layer vanishes due to very strong convective flows in the melt. When $\delta \rightarrow \infty$, the effective segregation coefficient approaches unity, that is, the incorporation of solute in the solid is equal to that in the melt.

A detailed discussion on microsegregation in the formalism of the boundary layer has been presented by Favier *et al.* [1982]. They have decoupled the convective and diffusive material transport and have assumed convection to be steady and diffusion to be time dependent. This assumption is interesting and quite questionable since it has been observed that unsteady convection is the cause of change of crystal radius.

Garandet *et al.* [1994] demonstrated that the results of Favier's and Wilson's models are quite close and also are in agreement with the theoretical model of Hurle *et al.* [1969]. But the segregation in lateral direction cannot be treated by a 1D model. Experimental results of Chung *et al.* [1997] clearly demonstrate that the solute concentration is a strong function of radial direction because of the strong convective effects in the melt and non-planar crystal-melt interface. Solute concentrations based on dynamic simulations from beginning of growth until the end have never been reported.

1.4 Scope of the Present Work

In summarizing the previous work on Conjugate Gradient methods, it has been realized that it is an extremely effective technique for solving linear systems when the coefficient matrix is symmetric positive definite. Generalization and efficient implementation of Conjugate Gradient technique for the linear systems with unsymmetric coefficient matrices is still an active area of research. Preconditioned Conjugate gradient (PCG) algorithm developed by Sheorey [2001] is one such generalization and present research is an assessment of this version of the PCG algorithm. The algorithm has been examined for the following test problems.

1. Two dimensional unsteady heat conduction in a region with variable thermal conductivity;
2. Two dimensional unsteady advection-diffusion problem of flow between two infinite parallel plates;
3. Momentum, energy and solutal transport equations arising in the modeling of Czochralski crystal growth.

Effect of the preconditioning technique used *i.e.*, incomplete lower-upper (ILU) preconditioning has been studied through the eigenvalue spectrum and condition numbers. In addition, solutal transport has been studied with a special interest in understanding solute segregation in the Czochralski crystal growth process.

1.5 Thesis Organization

The present thesis has been organized in the following manner:

1. Chapter 1 presents introduction, literature review and scope of the present research.
2. Chapter 2 presents mathematical background of the Conjugate Gradient method and then briefly summarizes various preconditioners.
3. Chapter 3 presents some numerical experiments with PCG.
4. Chapter 4 presents mathematical formulation for modeling transport phenomena in Czochralski crystal growth processes.

5. Chapter 5 gives a numerical methodology for solving the partial differential equations governing Czochralski crystal growth.

6. Chapter 6 carries conclusions of this study and scope of future work.

Chapter 2

Conjugate Gradient Algorithm: Mathematical Background and Preconditioning

In this chapter, the steepest descent method for solving linear systems is discussed. The powerful Conjugate Gradient(CG) algorithm is then presented as a variation on the Steepest Descent algorithm. The standard conjugate gradient algorithm (that is applicable only when the coefficient matrix is symmetric and positive definite) has been generalized to the case of any nonsingular coefficient matrix. The general CG algorithm thus obtained is then preconditioned with incomplete lower-upper(LU) decomposition of the coefficient matrix. Various preconditioners are finally summarized.

2.1 Descent Methods

If A is symmetric and positive definite then the problem of solving the linear system $Ax = b$ can be reformulated as a minimization problem. Define a function $J : \mathcal{R}^n \rightarrow \mathcal{R}$ by

$$J(y) = \frac{1}{2} y^T A y - y^T b \tag{2.1}$$

Then the vector that minimizes J is exactly the solution of $Ax = b$. The can be seen by writing the function J as

$$\begin{aligned} J(y) &= \frac{1}{2}y^TAy - y^TAx \\ &= \frac{1}{2}y^TAy - y^TAx + \frac{1}{2}x^TAx - \frac{1}{2}x^TAx \\ &= \frac{1}{2}(y-x)^TA(y-x) - \frac{1}{2}x^TAx \end{aligned} \tag{2.2}$$

The term $\frac{1}{2}x^TAx$ is independent of y, $J(y)$ is minimized exactly when $\frac{1}{2}(y - x)^TA(y-x)$ is minimized. Since A is positive definite, this term is positive unless $y - x = 0$. Thus it takes minimum value when and only when $y = x$.

Descent methods solve $Ax = b$ by minimizing J. These are iterative methods. Each descent method begins with an initial guess $x^{(0)}$ and generates a sequence of iterates $x^{(0)}$, $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ... such that at each step $J\left(x^{(k+1)}\right) \leq J\left(x^{(k)}\right)$, and preferably $J\left(x^{(k+1)}\right) < J\left(x^{(k)}\right)$. In this sense we get closer to the minimum at each step. If at some point we have $Ax^{(k)} = b$ or nearly so, we accept $x^{(k)}$ as the solution. The step from $x^{(k)}$ to $x^{(k+1)}$ has two ingredients: (i) choice of a search direction, and (ii) a line search in the chosen direction. Choosing a search line amounts to choosing a vector $p^{(k)}$ that indicates the direction in which we will travel to get from $x^{(k)}$ to $x^{(k+1)}$. Once a search direction has been chosen, $x^{(k+1)}$ will be chosen to be a point on the line $\left\{x^{(k)} + \alpha p^{(k)} \mid \alpha \in \mathcal{R}\right\}$. Thus we will have

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)} \tag{2.3}$$

for some real α_k. The process of choosing α_k from among all $\alpha_k \in \mathcal{R}$ is called line search. Here α_k is chosen is such a way that $J\left(x^{(k+1)}\right) \leq J\left(x^{(k)}\right)$. One way to ensure this is to choose α_k so that $J\left(x^{(k+1)}\right) = \min_{\alpha\in\mathcal{R}} J\left(x^{(k)} + \alpha p^{(k)}\right)$. This way of choosing α_k is called exact line search and it gives

$$\alpha_k = \frac{{p^{(k)}}^T r^{(k)}}{{p^{(k)}}^T A p^{(k)}} \tag{2.4}$$

where $r^{(k)} = b - Ax^{(k)}$.

2.1.1 Steepest Descent

The method of *steepest descent* takes $p^{(k)} = r^{(k)}$ and performs exact line searches. Since $r^{(k)} = -\nabla J(x^{(k)})$, the search direction is the direction of steepest descent of J from point $x^{(k)}$.

To search in the direction of the steepest descent is a quite natural idea but unfortunately, it does not work particularly well. Nevertheless, steepest descent is worth studying for at least two reasons: (i) It is a good vehicle for introducing the idea of preconditioning. (ii) Minor changes turns the steepest descent algorithm into the powerful conjugate-gradient method.

It is a simple matter to program the steepest descent algorithm. At each step approximate solution is updated using Equation 2.3 where values of α_k are calculated using Equation 2.4. Calculation of α_k requires, among all other things, multiplying the matrix A by the vector $p^{(k)}$. Cost of this operation depends on the degree of sparsity of A. In many applications the matrix-vector product is the most expensive step of the algorithm and the number of these steps needs to be minimized. Calculation of residual $r^{(k)} = b - Ax_{(k)}$ also seems to require an additional matrix-vector product $Ax_{(k)}$ and this can be avoided by a simple recursion formula

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)} \tag{2.5}$$

which is a consequence of Equation 2.3, to update the residual from one iteration to the next. Now the matrix-vector product $Ap^{(k)}$ need not be calculated again as it has been already calculated as part of computation of α_k. A prototype of steepest descent algorithm is given below (Watkins [2002]).

Algorithm 2.1: Prototype Steepest Descent Algorithm

$r^{(0)} = b - Ax^{(0)}$
$p^{(0)} = r^{(0)}$
for $k = 0, 1, 2, \ldots$
    $q^{(k)} = Ap^{(k)}$
    $\alpha_k = p^{(k)T} r^{(k)} / p^{(k)T} q^{(k)}$
    $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$

$$r^{(k+1)} = r^{(k)} - \alpha_k q^{(k)}$$
$$p^{(k+1)} = r^{(k+1)}$$

2.1.2 Geometric Interpretation and Convergence

The objective of the descent method is to minimize the function $J(y)$. From Equation 2.2 one can see that J is of the form

$$J(y) = \frac{1}{2}(y-x)^T A(y-x) - \gamma \tag{2.6}$$

where x is the solution of $Ax = b$, and γ is a constant. Since A is symmetric there exist an orthogonal matrix U such that $U^T AU$ is a diagonal matrix Λ. The main diagonal entries of Λ are eigenvalues of A, which are positive. Introducing new coordinates $z = U^T(y-x)$ and dropping the inessential constant γ, we see that minimizing $J(y)$ is equivalent to minimizing

$$\tilde{J}(z) = z^T \Lambda z = \sum_{i=1}^{n} \lambda_i {z_i}^2 \tag{2.7}$$

To get a picture of function $\tilde{J}$, consider a 2×2 case. Now $\tilde{J}$ is a function of two variables, so its contours or level surfaces $\tilde{J}(z_1, z_2) = c$ are curves in a plane which have the following form

$$\lambda_1 {z_1}^2 + \lambda_2 {z_2}^2 = c,$$

which are concentric ellipses centered on the origin. The orthogonal coordinate transformation $z = U^T(y-x)$ preserves lengths and angles, so the contours of J are also ellipses of the same shape. Such contours for the 2×2 case are shown in Figure 2.1.

The semiaxes of the ellipses are $\sqrt{c/\lambda_1}$ and $\sqrt{c/\lambda_2}$, whose ratio is $\sqrt{\lambda_2/\lambda_1}$. Eigenvalues of a positive definite matrix are the same as its singular values, so the spectral condition number is equal to the ratio of largest to smallest eigenvalue. Thus the shape of the contours depends on the condition number of A and greater the condition number, the more eccentric the ellipses are.

The dotted lines in Figure 2.1 represents the four steps of the steepest descent algorithm. For a given point, the search proceeds in the direction of the steepest descent, which is orthogonal to the contour line. The exact line search follows

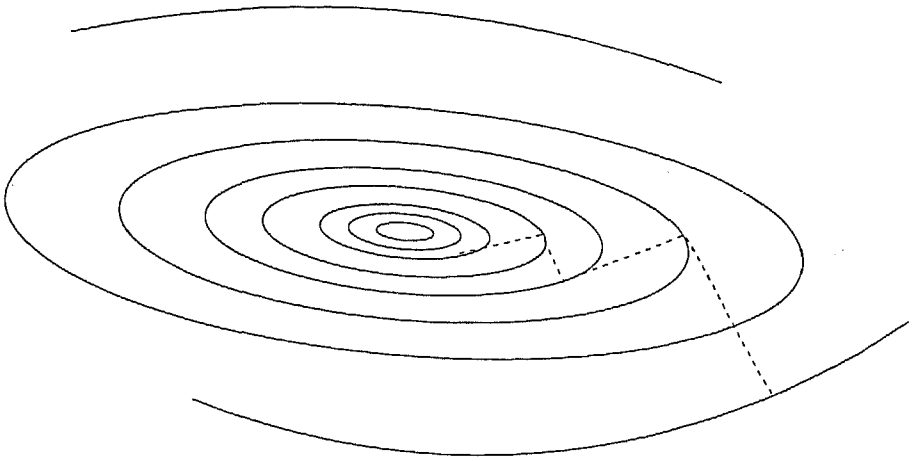


Figure 2.1: Steepest Descent in 2×2 case (Watkins [2002]).

the search line to the point at which J is minimized. J decreases as long as the search line cuts through the contours. The minimum occurs at the point at which the search line is tangent to a contour. Since the next search direction will be orthogonal to the contour at that point, each search direction is orthogonal to the previous one. Thus the search bounces back and forth in the canyon formed by the function $J(y)$ and proceeds steadily towards the minimum.

The minimum is quickly reached if A is well conditioned. In the best case, $\lambda_1 = \lambda_2$, the contours will be circles, the direction of steepest descent (from any starting point) points directly to the center, and the exact minimum is reached in one iteration. If A is well conditioned, contours will be nearly circular and the direction of steepest descent will point close to the center, and the method will converge rapidly. If, on the other hand, A is somewhat ill conditioned, the contours will be highly eccentric ellipses. From a given point, the direction of steepest descent is likely to point nowhere near the center and it would not be a good search direction. In this case the function J forms a steep, narrow canyon and the algorithm bounces back and forth in this canyon. It takes very slow steps and approaches the minimum with agonizing slowness. This phenomenon does not require extreme ill conditioning. Even if the system is modestly ill conditioned and well worth solving from the standpoint of numerical accuracy, the convergence can be very slow.

2.2 The Conjugate Gradient Method

The steepest descent method discussed earlier is limited by its lack of memory. It only uses information about $x^{(k)}$ to get to $x^{(k+1)}$ and all information from earlier iterations is forgotten. The *Conjugate Gradient* method is a simple variation on steepest descent that performs better because it has a memory. Conjugate gradient updates the solution in each iteration by using

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)} \tag{2.8}$$

which is exactly identical to the steepest descent approach, Equation 2.3. This gives the following residual update

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)} \tag{2.9}$$

Computation of α is organized a bit differently from steepest descent, but the difference is cosmetic. The choice of search directions, although inspired by the steepest descent, makes the whole difference. In conjugate gradient search direction is updated as

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)} \tag{2.10}$$

where β_k is chosen such that $p^{(k+1)}$ is $A-$orthogonal (conjugate) to all the previous search directions and this condition gives the following expression for β_k

$$\beta_k = \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle} \tag{2.11}$$

In addition to the orthogonality, derivation of the above expression uses self-adjoint property of A with respect to the inner product, Axelsson [1994].

After j iterations of the form $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$, both conjugate gradient and steepest descent algorithms give

$$x^{(j)} = \alpha_0 p^{(0)} + \alpha_1 p^{(1)} + ... + \alpha_{j-1} p^{(j-1)}$$

Thus $x^{(j)}$ lies in the space $\mathcal{S}_j$ spanned by the j search directions $p^{(0)}, ..., p^{(j-1)}$. This is true for both the algorithms. However, they pick different search directions. Interestingly, the two different sets of the search directions span the same space, $\mathcal{S}_j = \text{span}\{b, Ab, A^2b, ..., A^{j-1}b\}$, a Krylov subspace. Since steepest descent does exact line searches, $x^{(j)}$ minimizes the function J over the j lines $x^{(k)} + \alpha p^{(k)}$, $k = 0, ..., j-1$. The union of these lines is a tiny subset of $\mathcal{S}_j$. But, because of the conjugate search directions, conjugate gradient manages to pick out the $x^{(j)}$ that minimizes J over the entire subspace $\mathcal{S}_j$. This is called subspace minimization and therefore, conjugate gradient is also called a subspace minimization method.

Standard conjugate gradient algorithm can now be summarized as follows (Barrett *et al.* [1993]).

Algorithm 2.2: Standard Conjugate Gradient Algorithm

1. Set $k = 0$. Choose an initial approximation x_0. Compute $r^{(0)} = b - Ax^{(0)}$ and set $p_0 = r_0$.

2. Compute

$$\alpha_k = \langle r_k, r_k \rangle / \langle p_k, Ap_k \rangle$$

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$

$$\beta_k = \langle r^{(k+1)}, r^{(k+1)} \rangle / \langle r^{(k)}, r^{(k)} \rangle$$

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$$

3. Set $k = k + 1$

4. Repeat step$-$(2) and step$-$(3) until convergence in x is reached.

The method became popular due to many factors:

1. It has an optimality property over the relevant solution space, which usually means convergence to an acceptable accuracy with far fewer steps than the

number required for the finite termination property— *i.e.*, it has a relatively high rate of convergence.

2. The rate of convergence can be much improved with various preconditioning techniques.

3. The method is parameter free—*i.e.*, the user is not required to estimate any adjustable parameters.

4. The short recurrence relation makes the execution time per iteration and memory requirements acceptable.

5. The round-off error properties are acceptable.

2.3 Generalized Conjugate Gradient Method

With standard conjugate gradient method for solving $Ax = b$, Algorithm 2.2, success is guaranteed only when A is symmetric and positive definite. After k iterations of this algorithm solution $x^{(k)}$ can be written in the following form

$$x^{(k)} = x_0 + \sum_{i=1}^{k} \alpha_i A^{i-1}(Ax_0 - b) \tag{2.12}$$

and the algorithm chooses α_i's to minimize $\|x_k - x\|$, where $\|z\| = \langle z, Az\rangle^{1/2}$.

The standard conjugate gradient algorithm is modified to a generalized conjugate gradient algorithm that applies to any nonsingular coefficient matrix A. The trick used is to solve a modified system $A^TAx = A^Tb$ and since, in this modified system coefficient matrix is symmetric, standard conjugate gradient becomes applicable. Applying the standard CG algorithm to this modified system is equivalent to expanding in powers of (A^TA) instead of powers of A in the right hand side of the Equation 2.12 and this makes us able to minimize $x_k - x$ in Euclidean norm. For the new generalized algorithm we have the following equation.

$$x^{(k)} = x_0 + \sum_{i=1}^{k} \alpha_i (A^TA)^{i-1}A^T(Ax_0 - b) \tag{2.13}$$

Generalized conjugate gradient algorithm can now be summarized as follows (Kershaw [1978]).

Algorithm 2.3: Generalized Conjugate Gradient Algorithm

1. Set $k = 0$. Choose an initial approximation x_0. Compute $r^{(0)} = b - Ax^{(0)}$ and set $p_0 = A^T r_0$.

2. Compute

$$\alpha_k = \langle r_k, r_k \rangle / \langle p_k, p_k \rangle$$

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$

$$\beta_k = \langle r^{(k+1)}, r^{(k+1)} \rangle / \langle r^{(k)}, r^{(k)} \rangle$$

$$p^{(k+1)} = A^T r^{(k+1)} + \beta_k p^{(k)}$$

3. Set $k = k + 1$

4. Repeat step$-$(2) and step$-$(3) until convergence in x is reached.

2.4 Preconditioned Conjugate Gradient Method

Consider a linear system of equations,

$$Ax = b \tag{2.14}$$

where A is any nonsingular sparse matrix. The fact that one can always do an incomplete lower-upper (LU) decomposition of the coefficient matrix A is the motivation for the selection of incomplete LU-factorization as preconditioner. Exact decomposition will be subject to *fill-in* and moreover, that is almost equivalent to solving the linear system itself. So, instead of doing complete LU-decomposition, L is defined to have non-zero entries only within the semi bandwidth of A corresponding to the lower triangular portion of A and similarly U corresponding to the upper triangular portion of A. Thus the factorization is clearly approximate. The form of LU-decomposition algorithm in which all the diagonal elements of L are 1 (Crout's Decomposition) has been used. This incomplete LU-decomposition

algorithm is summarized below.

Algorithm 2.4: Crout's LU-decomposition algorithm

1. Choose element $a_{i,j}$ row wise, within the band of A.

2. If $a_{i,j}$ lies under the main diagonal, *i.e.*, $i > j$

$$l_{i,j} = a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} u_{k,j}$$

3. If $a_{i,j}$ lies on the main diagonal, *i.e.*, $i = j$

$$l_{i,i} = a_{i,i} - \sum_{k=1}^{i-1} l_{i,k} u_{k,i}$$

$$u_{i,i} = 1$$

4. If $a_{i,j}$ lies above the main diagonal, *i.e.*, $i < j$

$$u_{i,j} = {l_{i,i}}^{-1} \left(a_{i,j} - \sum_{k=1}^{i-1} l_{i,k} u_{k,j} \right)$$

Thus one can obtain an approximate inverse $(LU)^{-1}$ for A and then, can rewrite the original system, Equation 2.14, in the following form.

$$L^{-1} A U^{-1} (Ux) = L^{-1} b \tag{2.15}$$

Now, the generalized conjugate gradient method, Algorithm 2.3, when applied to the above preconditioned system we get the following *Preconditioned Conjugate Gradient Algorithm* (Kershaw [1978]).

Algorithm 2.5: Preconditioned Conjugate Gradient Algorithm

1. Set $k = 0$. Choose an initial approximation x_0. Compute $r^{(0)} = b - Ax^{(0)}$ and set $p_0 = (U^T U)^{-1} A^T (L L^T)^{-1} r_0$.

2. Compute

$$\alpha_k = \langle r_k, (LL^T)^{-1} r_k \rangle / \langle p_k, U^T U p_k \rangle$$

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$

$$\beta_k = \langle r^{(k+1)}, (LL^T)^{-1} r^{(k+1)} \rangle / \langle r^{(k)}, (LL^T)^{-1} r^{(k)} \rangle$$

$$p^{(k+1)} = (U^T U)^{-1} A^T (LL^T)^{-1} r^{(k+1)} + \beta_k p^{(k)}$$

3. Set $k = k + 1$

4. Repeat step−(2) and step−(3) until convergence in x is reached.

This preconditioned conjugate gradient is general and works for the case of any nonsingular coefficient matrix. Nevertheless, in the cases where the coefficient matrix is quite ill-conditioned there are chances of $U_{i,i}$ becoming zero during incomplete LU-decomposition and if so happen, algorithm will break down immediately. In such a case, one can simply set $U_{i,i}$ to a nonzero value and go on with the algorithm. Although this will introduce some error but LU-decomposition is approximate anyway.

2.5 Preconditioners

As discussed earlier, the convergence rate of iterative methods depends on spectral properties of the coefficient matrix. Hence one may attempt to transform the linear system into one that is equivalent in the sense that it has the same solution, but has more favorable spectral properties. A *preconditioner* is a matrix that affects such a transformation (Barrett *et al.* [1993]).

For instance, if M approximates the coefficient matrix A in some way, the transformed system

$$M^{-1} A x = M^{-1} b \tag{2.16}$$

has the same solution as the original system $Ax = b$, but the spectral properties of its coefficient matrix $M^{-1}A$ may be more favorable.

In devising a preconditioner, one is forced with a choice between finding a matrix M that approximates A, and for which solving a linear system a easier than solving one with A, or finding a matrix M that approximates A^{-1}, so that only multiplication by M is needed. The majority of available preconditioners fall in the first category.

2.5.1 Cost trade-off

Since using a preconditioner in an iterative method incurs some extra cost, both initially for the setup, and per iteration for applying it, there is a trade-off between the cost of constructing and applying the preconditioner, and the gain in convergence rate. Certain preconditioners little or no construction phase at all (for instance the SSOR preconditioner), but for others, such as incomplete factorizations, there can be substantial work involved.

On parallel machines there is a further trade-off between the efficacy of a preconditioner in the classical sense, and its parallel efficiency. Most of the traditional preconditioners have a large sequential component.

2.5.2 Left and Right Preconditioning

The transformation of linear system $A \rightarrow M_{-1}A$ is often not what is used in practice. For instance, the matrix $M_{-1}A$ is not symmetric, so, even if A and M are, the standard conjugate gradient method is immediately not applicable to this system. In this case, a way of deriving the preconditioned conjugate gradient method would be to split the preconditioner as $M = M_1M_2$ and to transform the system as

$$M_1{}^{-1}AM_2{}^{-1}(M_2x) = M_1{}^{-1}b \tag{2.17}$$

If M is symmetric and $M_1 = M_2{}^T$, it is obvious that now one has a method with a symmetric iteration matrix and hence, standard conjugate algorithm can be applied.

There is one more approach to preconditioning, which is easier to derive. Again consider the system in Equation 2.17. Here M_1 and M_2 are called the *left-* and *right preconditioners*, respectively. and one can simply apply an unpreconditioned iterative method to this system. Only two additional actions $r_0 = M_1{}^{-1}r_0$ before the iterative process and $x_n = M_2{}^{-1}x_n$ after are necessary.

2.5.3 Jacobi Preconditioning

The simplest preconditioner consists of just the diagonal of the matrix:

$$m_{i,j} = \begin{cases} a_{i,j} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

This is known as point incomplete preconditioner. It is possible to use this preconditioner without using any extra storage beyond that of the matrix itself. However, division operations are usually quite costly, so in practice storage is allocated for the reciprocals of the matrix diagonal.

Block Jacobi Methods

Block versions of the Jacobi preconditioner are derived by partitioning of the variables. If the index set $S = \{1, ..., n\}$ is partitioned as $S = \bigcup_i S_i$ with the sets S_i mutually disjoint, then

$$m_{i,j} = \begin{cases} a_{i,j} & \text{if } i \text{ and } j \text{ are in the same index subset} \\ 0 & \text{otherwise.} \end{cases}$$

The preconditioner is now a block-diagonal matrix.

Often, natural choices for the partitioning suggest themselves:

1. In problems with multiple physical variables per node, blocks can be formed by grouping the equations per node.

2. In structured matrices, such as those from partial differential equations on regular grids, a partitioning can be caused on the physical domain. Examples are a partitioning along lines in a 2D case, or planes in 3D case.

3. On parallel computers it is natural to let the partitioning coincide with the division of variables over the processors.

Jacobi preconditioners need very little storage, even in the block case, and they are easy to implement. Additionally, on parallel computers they do not present any particular problems. On the other hand, more sophisticated preconditioners usually yield a larger improvement.

2.5.4 SSOR Preconditioning

The SSOR (Symmetric Successive Over-Relaxation) like the Jacobi preconditioner, can be derived from the coefficient matrix without any work. If the original symmetric matrix is decomposed as

$$A = D + L + L^T$$

in its diagonal, lower, and upper triangular part, the SSOR matrix is defined as

$$M = (D + L)D^{-1}(D + L)^T,$$

or, parameterized by ω

$$M(\omega) = \frac{1}{2-\omega}\left(\frac{1}{\omega}D + L\right)\left(\frac{1}{\omega}D\right)^{-1}\left(\frac{1}{\omega}D + L\right)^T.$$

The optimal value of the ω parameter, like SOR method, will reduce the number of iterations to a lower order. Specifically, for second order elliptic problems a spectral condition number $\kappa({M_{\omega opt}}^{-1}A) = O(\sqrt{\kappa(A)})$ is attainable, Axelsson *et al.* [1984]. In practice, however, the spectral information is needed to calculate optimal ω.

The SSOR matrix is given in factored form, so this preconditioner shares many properties of other factorization-based methods. For instance, its suitability for vector processors or parallel architectures depends strongly on the ordering of variables. On the other hand, since this factorization is given a *priori*, there is no possibility of breakdown as in incomplete factorization methods.

2.5.5 Incomplete Factorization Preconditioners

A broad class of preconditioners is based on incomplete factorizations of the coefficient matrix. The factorization is called incomplete if during the factorization process certain *fill-in* elements, nonzero elements in the factorization in positions where the original matrix had a zero, have been ignored. Such a preconditioner is then given a factored form $M = LU$ with L lower and U upper triangular. The efficacy of the preconditioner depends on how well M^{-1} approximates A^{-1}.

Unlike the Jacobi and SSOR preconditioners, incomplete factorization preconditioners need a non-trivial creation stage. Incomplete factorizations may break

down (attempted division by zero pivot) or result in indefinite matrices (negative pivots) even if the full factorization of the same matrix is guaranteed to exist and yield a positive definite matrix.

An incomplete factorization is guaranteed to exist for many factorization strategies if the original matrix is an $M-$matrix (*i.e.*, $M_{i,j} \leq 0$ for all $i \neq j$). This was originally proved by Meijerink and Van der Vost [1981]. In cases where the pivots are zero or negative, strategies have been proposed such as substituting an arbitrary positive number (Kershaw [1978]), or restarting the factorization on $A + \alpha I$ for some positive value of α (Manteuffel [1980]).

An important consideration for incomplete factorization preconditioners is the cost of the factorization process. Even if the incomplete factorization exists, the number of operations involved in creating it is at least as much as for solving a system with such a coefficient matrix, so the cost may equal that of one or more iterations of the iterative method. On parallel computers this problem is aggravated by the general poor parallel efficiency of the factorization.

Point Incomplete Factorizations

The most common type of incomplete factorization is based on taking on a set S of matrix positions, and keeping all positions outside this set equal to zero during factorization. The resulting factorization is incomplete in the sense that fill is suppressed.

The set S is usually chosen to encompass all positions (i, j) for which $a_{i,j} \neq 0$. A position that is zero in A but not so in an exact factorization is called a *fill* position, and if it is outside S, the fill there is said to be discarded. Often S is chosen to coincide with the set of nonzero positions in A, discarding all fill. There are two major strategies for accepting or discarding fill-in, one structural and one numerical. The structural strategy is that of accepting fill-in only to a certain level. The numerical fill strategy is that of drop tolerances: fill is ignored if it is too small, for a suitable definition of small. Although this definition makes more sense mathematically, it is harder to implement in practice, since the amount of storage needed for the factorization is not easy to predict.

Block Incomplete Factorizations

The starting point for an incomplete block factorization is a partitioning of the matrix. Then an incomplete factorization is performed using the matrix blocks as basic entities. The most important difference with the point methods arises in the inversion of the pivot blocks. Whereas inverting a scaler is easily done, in block case two problems arise. First, inverting the pivot block is likely to be a costly operation. Second, initially the diagonal blocks of the matrix are likely to be sparse and one would like to maintain this structure throughout the factorization. Hence the need for approximation of inverses arises.

In block factorization a pivot block is generally forced to be sparse, typically of banded form, and that we need an approximation to its inverse that has similar structure. Further this approximation should be easily computable, so the option of calculating the full inverse is generally ruled out by taking the banded part of it. For example, the simplest approximation to A^{-1} is the diagonal matrix D of the reciprocals of the diagonal of A.

Banded approximations to the inverse of banded matrices have a theoretical justification. In the context of partial differential equations the diagonal blocks of the coefficient matrix are usually diagonally dominant. For such matrices, the elements of the inverse have a size that is exponentially decreasing in their distance from the main diagonal.

2.5.6 Polynomial Preconditioners

Polynomial preconditioners can be considered as the second class of preconditioners where the preconditioning matrix is a direct approximation of the inverse of the coefficient matrix. Suppose that the coefficient matrix A of the linear system is normalized to the form $A = I - B$, and that the spectral radius of B is less than one. Then using the Neumann series, one can write the inverse of A as $A^{-1} = \sum_{k=0}^{\infty} B^k$ and thus, an approximation may be derived by truncating this infinite series.

Dubois *et al.* [1979] investigated the relationship between a basic method using a splitting $A = M - N$, and a polynomially preconditioned method with

$$M_p{}^{-1} = \left(\sum_{i=0}^{p-1} \left(I - M^{-1}A \right)^i \right) M^{-1}.$$

The basic result id that for classical methods, k steps of the polynomially preconditioned method are exactly equivalent to kp steps of the original method. For accelerated methods, especially the Chebyshev method, the preconditioned iteration can improve the number of iterations by at most a factor of p.

Although there is no gain in the number of times the coefficient matrix is applied, polynomial preconditioning does eliminate a large fraction of the inner products and update operations and there may be an overall increase in efficiency.

Chapter 3

Numerical Experiments with PCG

In this chapter, numerical experiments with the Preconditioned Conjugate Gradient algorithm have been presented. A nonlinear unsteady heat conduction problem is first considered and a study of variation in CPU time with the size of the problem is done for three linear system solvers, namely Gaussian Elimination, Gauss-Seidel and Preconditioned Conjugate Gradient. In the second test case, the suitability of Preconditioned Conjugate Gradient for solving advection-diffusion equation is assessed. To study the effect of preconditioning, eigenvalues and condition numbers of the original and preconditioned matrices have been studied for the advection-diffusion problem.

3.1 Heat Conduction Problem

Here the following dimensionless equation governing unsteady heat conduction in a two dimensional variable conductivity region (Figure 3.1) is solved.

$$\frac{\partial}{\partial X}\left(k(\theta)\frac{\partial \theta}{\partial X}\right)+\frac{\partial}{\partial Y}\left(k(\theta)\frac{\partial \theta}{\partial Y}\right)=\frac{\partial \theta}{\partial \tau} \tag{3.1}$$

Thermal conductivity in this region varies in the following manner:

$$k(\theta)=1-\beta\theta \tag{3.2}$$

and the values of β that has been considered are $\beta = 0$ and 0.4.

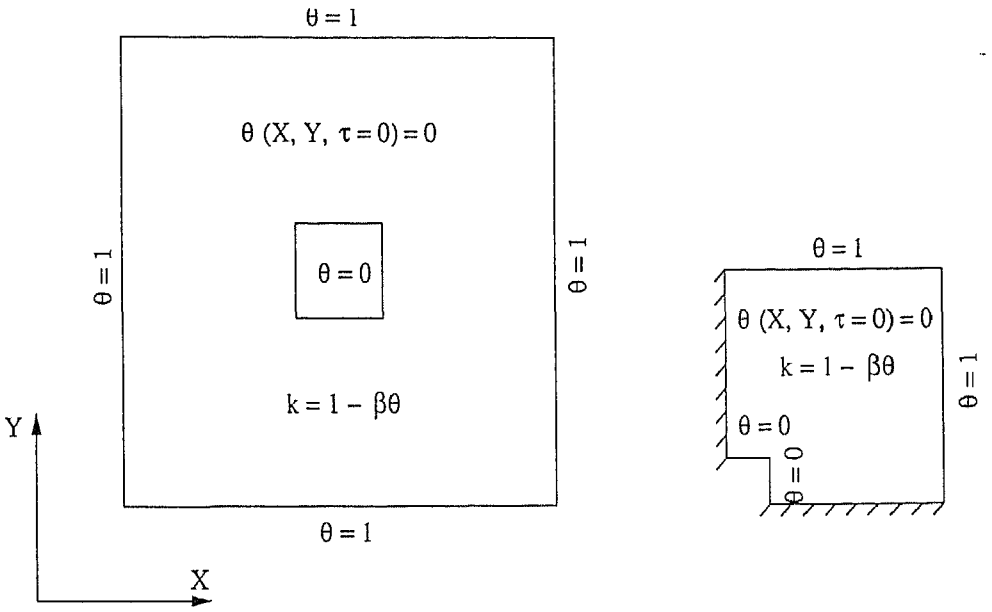


Figure 3.1: Physical and Computational domains for the nonlinear conduction problem.

Physical and computational domains along with initial and boundary conditions are shown in Figure 3.1. Governing Equation 3.1 is discretized using pure implicit scheme for the unsteady term and second order central difference for all other terms. For a point (i, j) in the computational domain this discretization gives following relation for temperatures between two consecutive time steps:

$$\begin{aligned} s\left(k_{i+1,j}^{(p+1)} - k_{i-1,j}^{(p+1)} - 4k_{i,j}^{(p+1)}\right)\theta_{i-1,j}^{(p+1)} + s\left(k_{i,j+1}^{(p+1)} - k_{i,j-1}^{(p+1)} - 4k_{i,j}^{(p+1)}\right)\theta_{i,j-1}^{(p+1)} \quad (3.3)\\ +s\left(k_{i,j-1}^{(p+1)} - k_{i,j+1}^{(p+1)} - 4k_{i,j}^{(p+1)}\right)\theta_{i,j+1}^{(p+1)} + s\left(k_{i-1,j}^{(p+1)} - k_{i+1,j}^{(p+1)} - 4k_{i,j}^{(p+1)}\right)\theta_{i+1,j}^{(p+1)}\\ + \left(1 + 16sk_{i,j}^{(p+1)}\right) = \theta_{i,j}^{(p)} \end{aligned}$$

Where $s = \frac{\Delta\tau}{4h^2}$ and $h(= \Delta X = \Delta Y)$ is the discretization in both X and Y directions.

3.1.1 Solution Procedure

The solution algorithm has been used to obtain the time marching temperature field inside the computational domain.

1. Set time step $p = 0$ and initialize the temperature field at the p^{th} time step, $\theta^{(p)}$, with a given initial temperature distribution.

2. Use the following steps to calculate temperature at $(p+1)^{th}$ time step, $\theta^{(p+1)}$, by solving the nonlinear system of equations represented by the discretized governing Equation 3.4.
   (a) Initialize $\theta^{(p+1)} = \theta^{(p)}$
   (b) Evaluate k values in Equation 3.4 using $k^{(p+1)} = k\left(\theta^{(p+1)}\right)$ and thus generate a linear system of equations in $\theta^{(p+1)}$
   (c) Solve the system of linear equations for $\theta^{(p+1)}$ using a linear system solver
   (d) Repeat steps (b) and (c) until convergence in k and $\theta^{(p+1)}$ is achieved

3. Set $\theta^{(p)} = \theta^{(p+1)}$ and $p = p+1$.

4. Repeat step-2 and step-3 until steady state is reached.

3.1.2 Results and Discussion

The results reported for the nonlinear conduction problem includes

1. Comparison of CPU time for solving the problem till steady-state for three solvers namely Gaussian elimination, Gauss-Seidel iterations and preconditioned conjugate gradient method.

2. Temperature contours obtained using the above linear system solvers.

Results have been presented for two variations in thermal conductivity, *i.e.*, $\beta = 0$ and 0.4. During nonlinearity iterations (l) the convergence criteria used is

$$\frac{\left|\theta_i^{l+1} - \theta_i^{(l)}\right|}{\left|\theta_i^{(l+1)}\right|} \times 100 < 0.01$$

for all grid points (i) in the computational domain.

CPU time variation shown in Figure 3.2 clearly shows that PCG is much faster then Gaussian elimination and Gauss Seidel. For example, for $\beta = 0.4$ variation in thermal conductivity, problem solving with PCG is $10-20$ times cheaper (in terms of CPU time) than Gauss-Seidel and almost 1900 times cheaper than Gaussian elimination. Moreover, it has been generally observed that PCG becomes a

more attractive choice as the number of grid points in the computational domain increases.

Figure 3.3 shows temperature contours inside the 2D domain for a linear ($\beta = 0$) and nonlinear ($\beta = 0.4$) case. In both the cases, temperature fields obtained using three different solvers are identical. This verifies the uniqueness of inverses for the matrices arising in this problem with respect to different matrix inverters used.

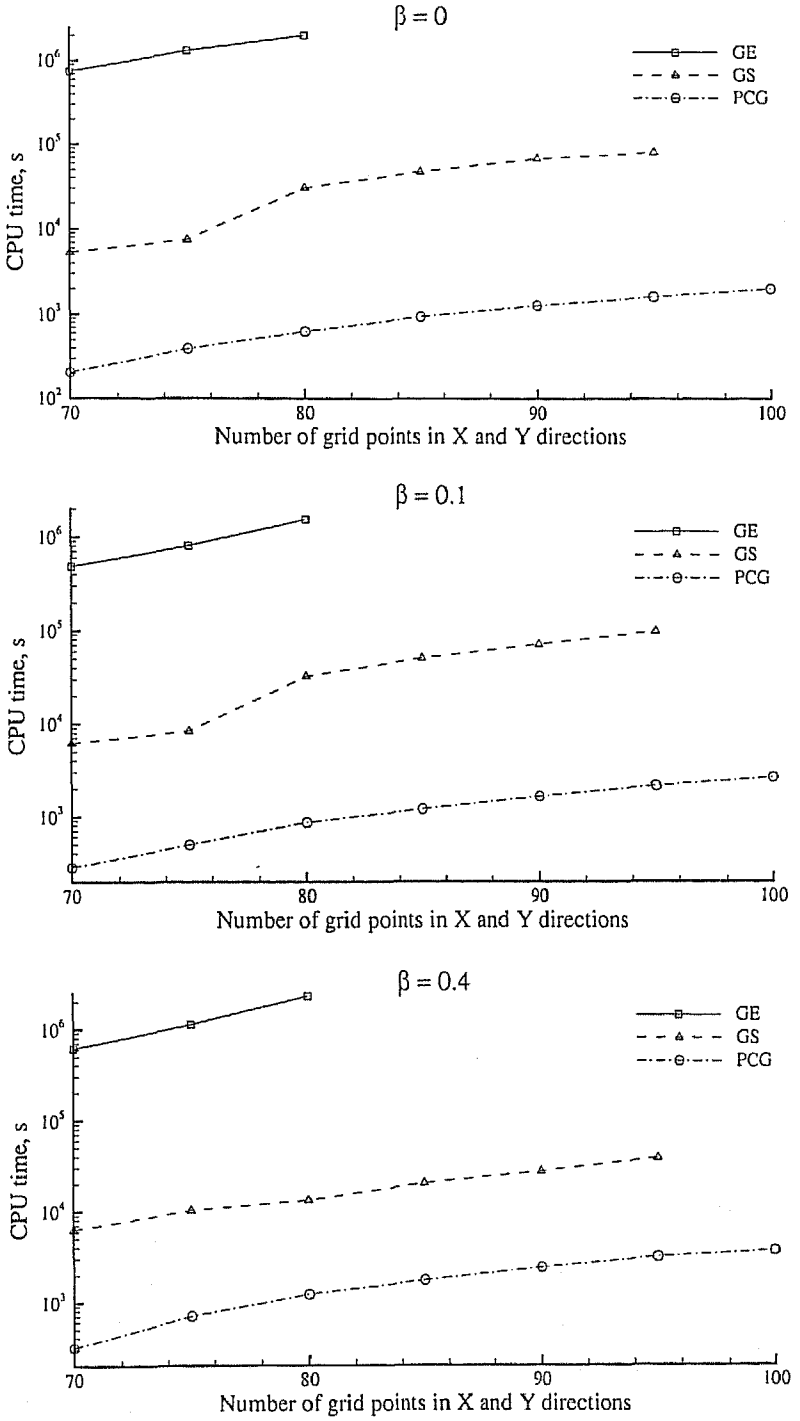


Figure 3.2: Variation in CPU time with number of grid points in X and Y directions. Thermal conductivity varies with temperature as $k = 1 - \beta\theta$.

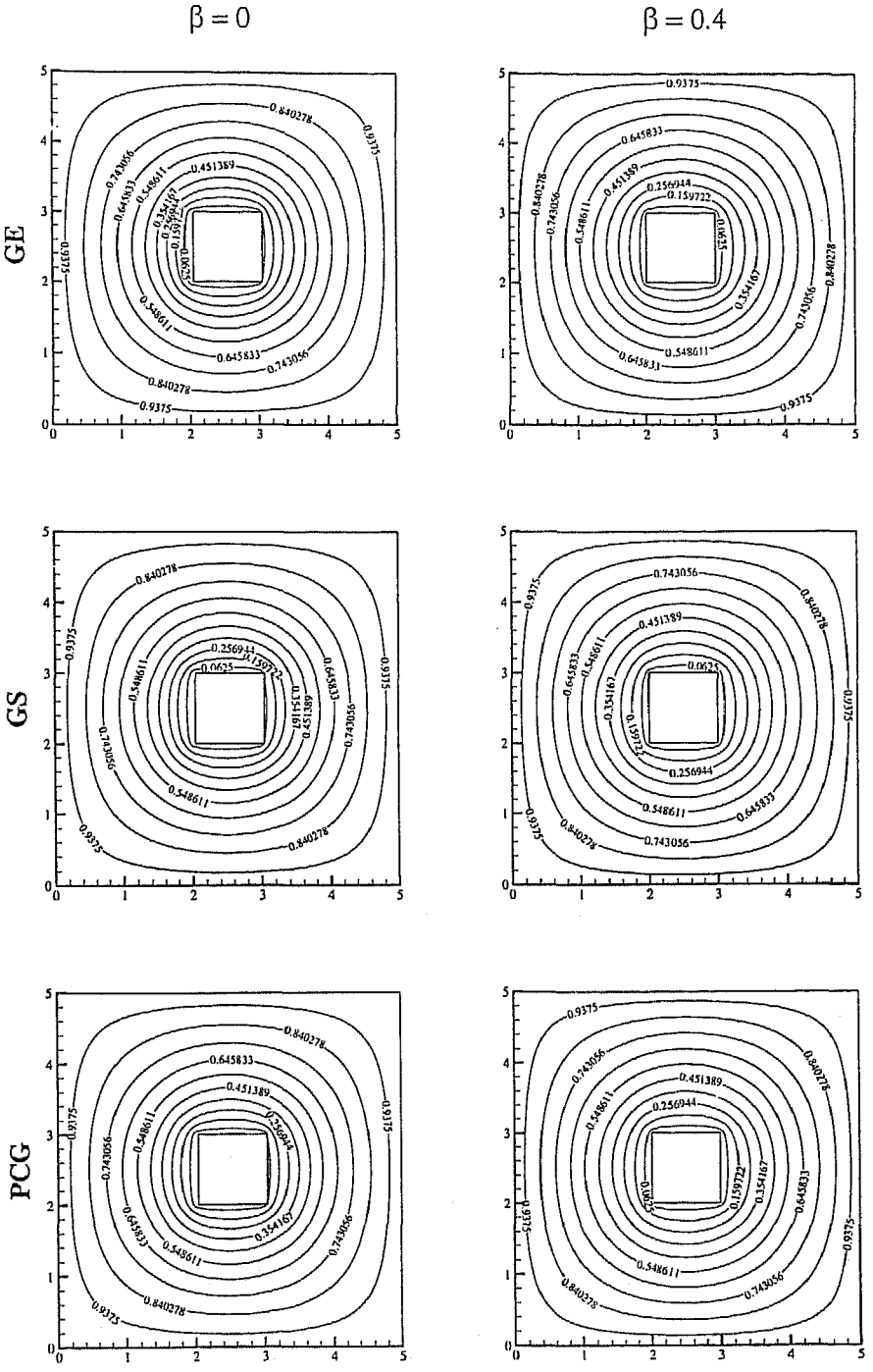


Figure 3.3: Temperature contours for linear ($\beta = 0$) and non linear ($\beta = 0.4$) conduction using three different matrix inverters. Thermal conductivity varies with temperature as $k = 1 - \beta\theta$.

3.2 Advection-Diffusion Problem

Here the numerical solution of the energy equation (advection-diffusion form) for laminar flow between two infinite parallel plates has been considered as the test problem. Flow is assumed to be hydrodynamically fully developed. Temperature of both the plates is kept constant at $T_i = 0$ for $x < 0$, and at $T_0 = 1$ for $x \geq 0$. Incoming fluid also has a temperature T_i. Physical domain for the problem is shown in the Figure 3.4.

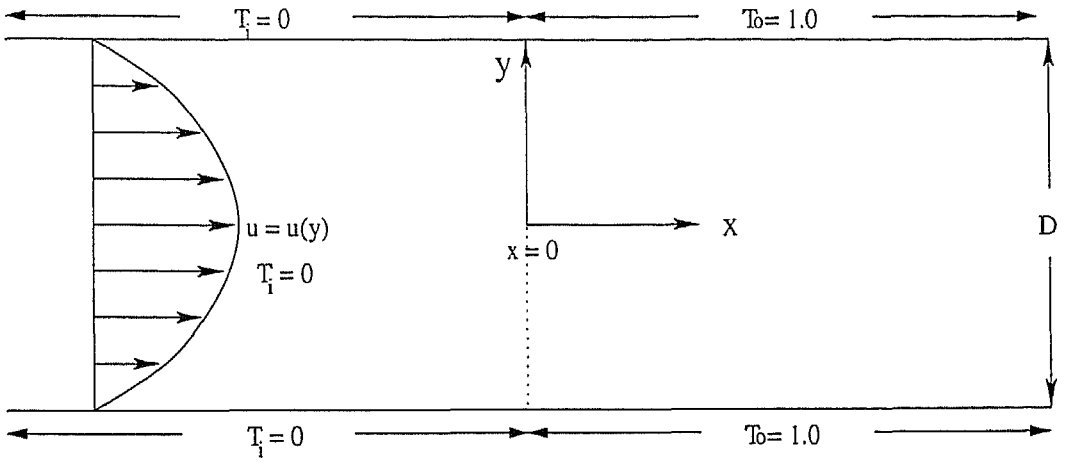


Figure 3.4: Laminar, hydro-dynamically fully developed flow between two infinite parallel plates with constant wall temperature conditions.

The velocity profile for this flow is parabolic and is given by the following equation.

$$u = \frac{3}{2}\left[1 - \left(\frac{y}{D/2}\right)^2\right] \tag{3.4}$$

Now with the assumptions :

1. constant thermal conductivity of fluid;
2. negligible viscous dissipation;
3. negligible compressibility effects.

the unsteady energy equation for this flow is as follows (Bejan [1984]).

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{3.5}$$

3.2.1 Nondimensionalization

Introducing the following dimensionless quantities:

$$\theta = \frac{T - T_i}{T_0 - T_i}, \quad X = \frac{x}{D/2}, \quad Y = \frac{y}{D/2}, \quad \tau = \frac{t}{\frac{D/2}{U}}, \quad and \quad u^* = \frac{u}{U}$$

where

T_i : incoming fluid temperature

T_0 : wall temperature

D : distance between two plates

U : average axial velocity.

The governing Equations 3.4 and 3.5 takes the following form.

$$u^* = \frac{3}{2}\left[1 - Y^2\right] \tag{3.6}$$

$$\frac{\partial \theta}{\partial \tau} + \frac{\partial (u^*\theta)}{\partial X} = \frac{1}{\mathrm{Pe}}\left(\frac{\partial^2 \theta}{\partial X^2} + \frac{\partial^2 \theta}{\partial Y^2}\right) \tag{3.7}$$

where

$$\mathrm{Pe} = \frac{UD/2}{\alpha}.$$

3.2.2 Initial and Boundary Conditions

The physical domain, Figure 3.4 has geometric, hydrodynamic and thermal symmetry about the mid plane (the plane that contains $x-$axis and is parallel to plates). Therefore, the portion above this plane is considered as computational domain.

Now for this computational domain the non-dimensional initial and boundary conditions are as follows.

Initial condition:

at $\tau = 0$, for all X and Y

$$\theta = 0$$

Boundary conditions:

for $\tau > 0$,

at the inlet of the computational domain ($X = 0$ and $0 \leq Y \leq 1$)

$$\theta = 0$$

at the outlet of the computational domain ($X \gg 1$ and $0 \leq Y \leq 1$)

$$\frac{\partial \theta}{\partial X} = 0$$

at the mid plane ($X \geq 0$ and $Y = 0$)

$$\frac{\partial \theta}{\partial Y} = 0$$

at the upper wall ($X \geq 0$ and $Y = 1$)

$$\theta = 1$$

The computational domain along with dimensionless boundary conditions is shown below.

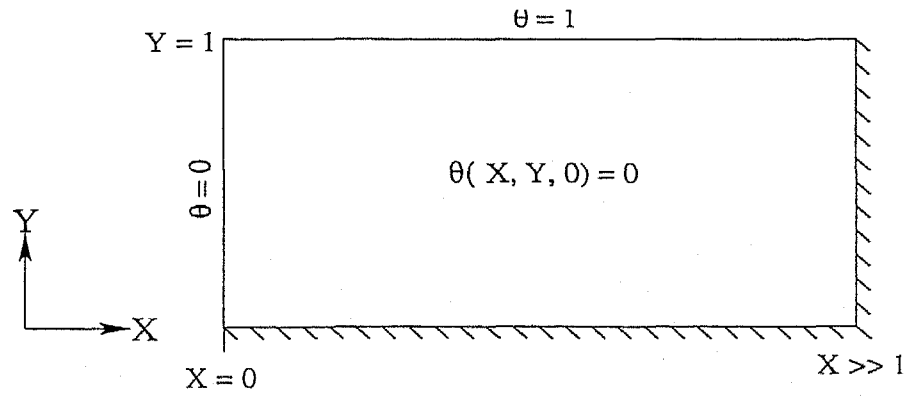


Figure 3.5: Computational domain for the advection-diffusion problem.

3.2.3 Discretization

Governing nondimensional equation, Equation 3.7, is discretized in the following manner.

Unsteady term : Pure implicit

Advection term : QUICK (In view of minimizing error due to false diffusion.)

Conduction terms : Second order central difference

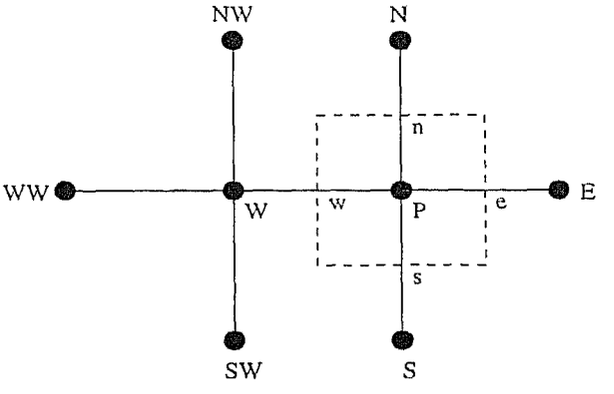


Figure 3.6: Computational molecule for discretization.

For this discretization, the present time-step is p and the next time-step is taken as $p+1$. A computational molecule for discretization is shown below in Figure 3.6. The unsteady term at time $p+1$, for the grid point P, can be written as :

$$\left(\frac{\partial\theta}{\partial\tau}\right)_P^{p+1} = \frac{\theta_P^{p+1} - \theta_P^p}{\Delta\tau} \tag{3.8}$$

The advection term at point P and time $p+1$ can be written as follows.

$$\left[\frac{\partial(u^*\theta)}{\partial X}\right]_P^{p+1} = \frac{(u^*\theta)_e^{p+1} - (u^*\theta)_w^{p+1}}{\Delta X} \tag{3.9}$$

Using QUICK, at any time value of advection flux from the face e of the control volume surrounding P can be written as.

$$(u^*\theta)_e = (u^*\theta)_{LIN_e} - \frac{1}{8}CURVN_e(\Delta X)^2 + \frac{1}{24}CURVT_e(\Delta Y)^2$$

where

$$(u^*\theta)_{LIN_e} = \frac{1}{2}\left[(u^*\theta)_P + (u^*\theta)_E\right]$$

for $u^* > 0$

$$CURVN_e = \frac{(u^*\theta)_E - 2\,(u^*\theta)_P + (u^*\theta)_W}{(\Delta X)^2}$$

and

$$CURVT_e = \frac{(u^*\theta)_N - 2\,(u^*\theta)_P + (u^*\theta)_S}{(\Delta Y)^2}$$

Similarly it can be written for the face w

$$(u^*\theta)_w = (u^*\theta)_{LIN_w} - \frac{1}{8}CURVN_w(\Delta X)^2 + \frac{1}{24}CURVT_w(\Delta Y)^2$$

where

$$(u^*\theta)_{LIN_w} = \frac{1}{2}\left[(u^*\theta)_W + (u^*\theta)_P\right]$$

$$CURVN_w = \frac{(u^*\theta)_P - 2\,(u^*\theta)_W + (u^*\theta)_{WW}}{(\Delta X)^2}$$

and

$$CURVT_w = \frac{(u^*\theta)_W\,N - 2\,(u^*\theta)_W + (u^*\theta)_{WS}}{(\Delta Y)^2}$$

Substituting the above values of $(u^*\theta)_e$ and $(u^*\theta)_w$ in Equation 3.9 gives

$$\begin{aligned}(\Delta X)\left[\frac{\partial(u^*\theta)}{\partial X}\right]_P^{p+1} &= \frac{1}{2}\left[(u^*\theta)_E^{p+1} - (u^*\theta)_W^{p+1}\right] \\ &-\frac{1}{8}\left[(u^*\theta)_E^{p+1} - 3(u^*\theta)_P^{p+1} + 3(u^*\theta)_W^{p+1} - (u^*\theta)_{WW}^{p+1}\right] \\ &+\frac{1}{24}\left[(u^*\theta)_N^{p+1} - 2(u^*\theta)_P^{p+1} + (u^*\theta)_S^{p+1} - (u^*\theta)_{WN}^{p+1} + 2(u^*\theta)_W^{p+1} - (u^*\theta)_{WS}^{p+1}\right]\end{aligned} \tag{3.10}$$

Discretization for conduction terms can be written as

$$\left(\frac{\partial^2\theta}{\partial X^2}+\frac{\partial^2\theta}{\partial Y^2}\right)_P^{p+1}=\left(\frac{\theta_W^{p+1}-2\theta_P^{p+1}+\theta_E^{p+1}}{(\Delta X)^2}+\frac{\theta_N^{p+1}-2\theta_P^{p+1}+\theta_S^{p+1}}{(\Delta Y)^2}\right) \quad (3.11)$$

Substituting the discretized values of various terms from Equations 3.8, 3.10 and 3.11 in the governing equation 3.7, the following algebraic equation is obtained.

$$\begin{aligned}&\left(3u_P^*\mathrm{Pe}_c\right)\theta_{WW}^{p+1}-\left(19u_P^*\mathrm{Pe}_c+24\right)\theta_W^{p+1}+\left[r+7u_P^*\mathrm{Pe}_c+48\left(1+R^2\right)\right]\theta_P^{p+1}\\&-\left(24-9u_P^*\mathrm{Pe}_c\right)\theta_E^{p+1}-\left(24R^2-u_N^*\mathrm{Pe}_c\right)\theta_N^{p+1}-\left(24R^2-u_S^*\mathrm{Pe}_c\right)\theta_S^{p+1}\\&-\left(u_N^*\mathrm{Pe}_c\right)\theta_{NW}^{p+1}-\left(u_S^*\mathrm{Pe}_c\right)\theta_{SW}^{p+1}=r\theta_P^p \quad (3.12)\end{aligned}$$

Here Pe_c is cell Peclet number, defined as $\mathrm{Pe}_c=\mathrm{Pe}(\Delta X)$, $R=\frac{\Delta X}{\Delta Y}$ and

$$r=\frac{24(\Delta X)\mathrm{Pe}_c}{\Delta\tau}.$$

3.2.4 Nusselt Number Calculation

To address the basic question of heat transfer for this flow, let us consider a relationship between the heat flux and wall-fluid temperature difference. Since fluid temperature varies over the cross-section, $\Delta T=T_0-T_m$ is conventionally selected as the the wall-fluid temperature difference. We are then interested in the heat transfer coefficient

$$h=\frac{k\left(\frac{\partial T}{\partial y}\right)_{wall}}{T_0-T_m} \quad (3.13)$$

Using the above heat transfer coefficient, Nusselt number can be defined as

$$\mathrm{Nu}=\frac{hD_h}{k} \quad (3.14)$$

where D_h is the hydraulic diameter ($D_h=2D$ in this case). The value of mean temperature, T_m in Equation 3.13, is given by

$$T_m=\frac{\int_{-D/2}^{+D/2}(uT)\,dy}{DU} \quad (3.15)$$

Introducing the previously used dimensionless quantities and using Equations 3.13, 3.14 and 3.15, expressions for Nusselt number are given by the following two equations:

$$\mathrm{Nu} = \frac{4\left(\frac{\partial\theta}{\partial Y}\right)_{Y=1}}{1-\theta_m} \tag{3.16}$$

and

$$\theta_m = \frac{3}{2}\int_0^1 \left[\left(1-Y^2\right)\theta\right] dy \tag{3.17}$$

3.2.5 Solution Procedure

Here Equation 3.12 represents the discretized governing equation and the initial and boundary conditions are as given in Section 3.2.2. Starting with a given initial condition for temperature, a marching solution in time for the entire domain has been found. It is clear from Equation 3.12, that a linear system of equations is to be solved in each time-step to proceed in this way. Preconditioned Conjugate Gradient (PCG) algorithm is used to solve this linear system at each time step. In this manner the temperature field is obtained as a function of time until steady-state is reached.

The Nusselt number is evaluated using Equations 3.16 and 3.17. Simpson's 1/3 rule has been used to calculate the bulk mean temperature given by equation 3.17.

3.2.6 Results and Discussion

The following results have been presented for this analysis:

1. Variation of Nusselt number with time at selected locations on the heated plate (Figure 3.7)

2. Steady-state variation of Nusselt number along the heated plate (Figure 3.8)

Both the results have been presented for six values of the Peclet number, *i.e.*, 0.1, 1, 10, 100, 500 and 1000. Figure 3.7 shows the variation of Nusselt number with time at selected $X-$locations along the plate for six different values of Peclet

number. Here, at any $X-$location Nusselt number decreases exponentially with time and then becomes constant at a steady-state value. This steady-state value decreases with the distance along the heated plate. This is expected since along the plate as the fluid gets heated, rate of plate heat transfer and therefore the Nusselt number decreases.

In Figure 3.8 both the plots depict the variation in steady-state Nusselt number with distance along the heated plates. The value of Nusselt number first decreases exponentially with distance and then becomes constant at a larger distance. At this distance the flow is said to be thermally developed. The value of Nusselt number for thermally developed flows has been reported in literature. For the considered problem, analytic solutions in literature reports Nu = 7.54. It has been observed that, when the flow is thermally developed, numerically calculated Nusselt number closely matches this value. Moreover, at higher values of Peclet number it exactly matches the reported value. An explanation for this fact is that, for this case, the analytic solution of energy equation reported in the literature is valid only for large values of Peclet number (Pe >> 1).

In Figure 3.8, it is to be noted that in each plot, the upper two $X-Y$ curves are slightly shifted in vertically upward direction and for these two, the values labeled on the $Y-$axis are slightly inconsistent The labels on $Y-$axis are exactly consistent with the bottom-most curve. Labels on the $X-$axis are consistent with all the curves in all the plots.

For solving the linear system of equations, three solvers namely, Preconditioned Conjugate Gradient (PCG), point-by-point Gauss-Seidel and Gaussian elimination have been used. Preconditioned Conjugate Gradient (PCG) has been applied in two different ways. In the first, incomplete lower-upper (ILU) decomposition is done in each time-step when the solver is used. In the second, incomplete lower-upper (ILU) decomposition is done only once when the solver is used for the very first time in the computer program. For this problem, Point by Point Gauss-Seidel emerges as the fastest solver in terms of CPU time. Performance of Preconditioned Conjugate Gradient (PCG), when implemented with incomplete LU decomposition in every time step is almost ten times slower than point-by-point Gauss-Seidel. When it is implemented only once, it is almost four times slower than point-by-point Gauss-Seidel. Performance of Gaussian elimination is awfully slow (*i.e.*, almost more than hundred times slower than point-by-Point

Gauss-Seidel). Moreover, since it is a linear problem the effectiveness of PCG is not clearly seen. The CPU time comparision is expected to show in favourable light for nonlinear problems.

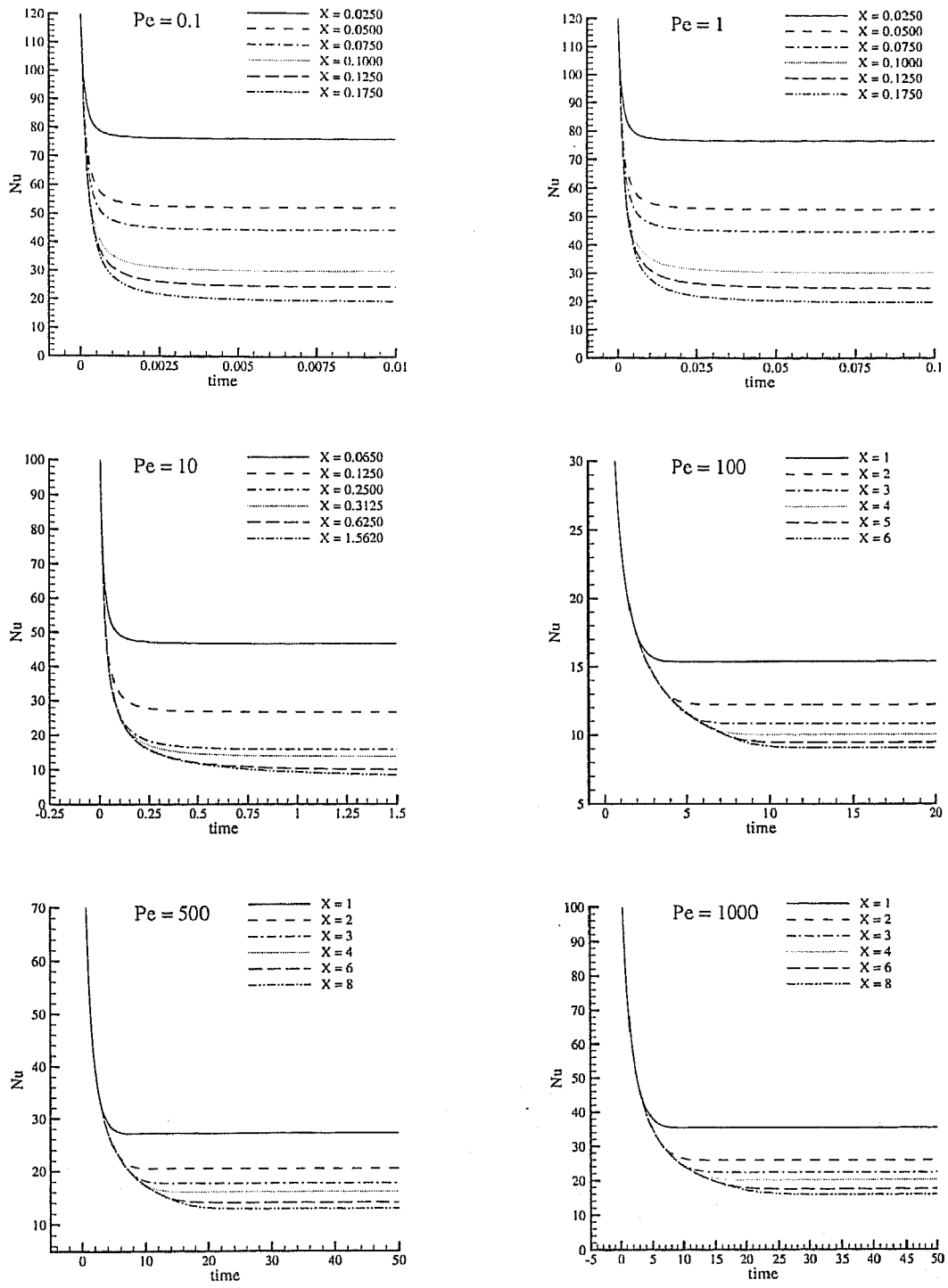


Figure 3.7: Variation of Nusselt number with time at selected $X-$locations, for different values of Peclet number. Here X is the distance along the heated wall.

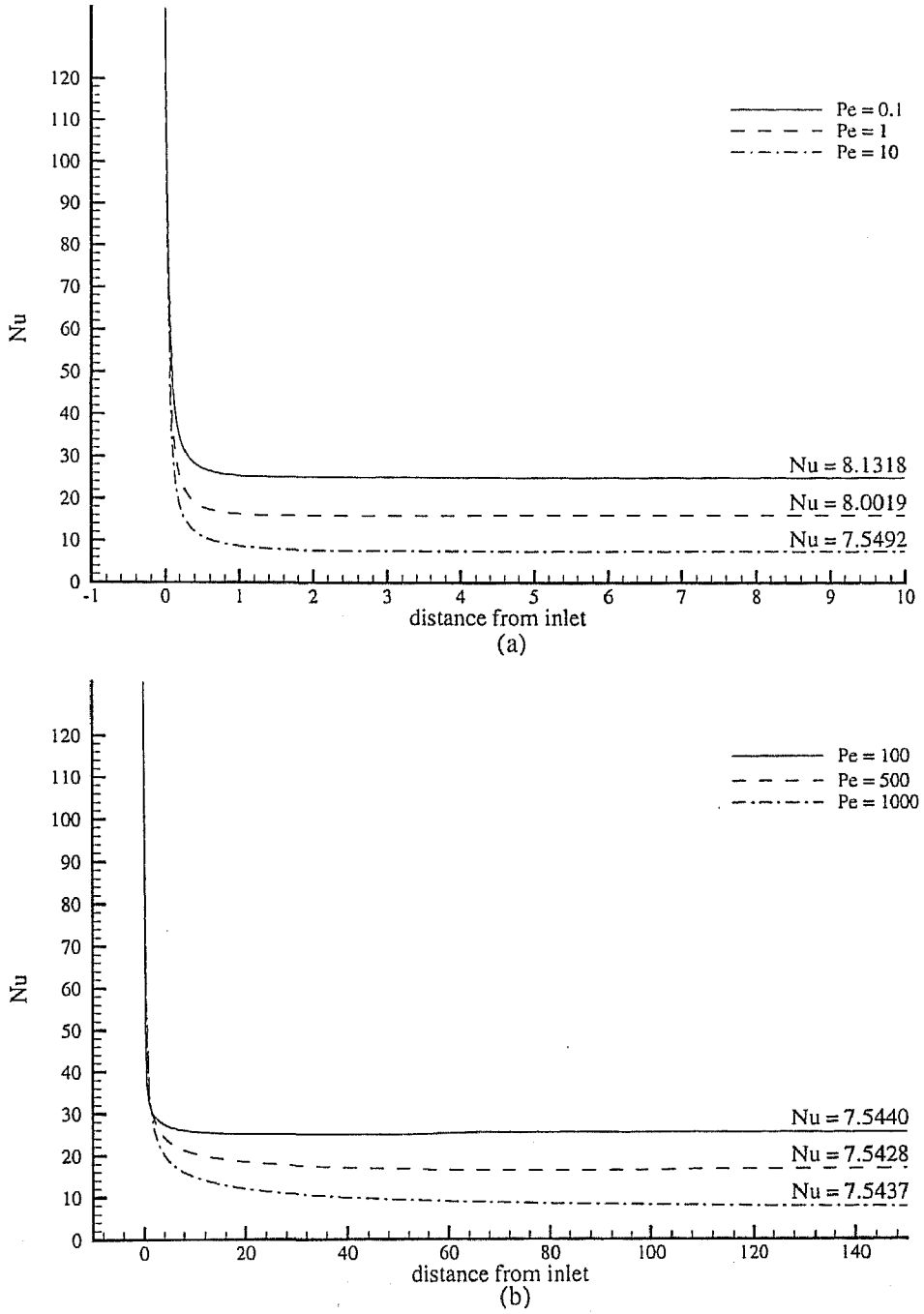


Figure 3.8: Steady-state variation of Nusselt number with distance along the heated plate.

3.3 Assessment of Eigenvalues and Condition Numbers

3.3.1 Eigenvalues

As discussed earlier in Section 2.5, the spread of eigenvalues of the coefficient matrix is a qualitative measure of the intrinsic difficulty in solving a linear system. Thus the eigenvalue spectrum provides a qualitative basis for comparing different coefficient matrices. Moreover, the effect of preconditioning can be seen by comparing the eigenvalue spectra of the preconditioned matrix with the original coefficient matrix.

3.3.2 Condition Numbers

Another important issue in solving linear system $Ax = b$ is the sensitivity of the solution for small perturbations in b and *condition number* of the coefficient matrix A is a measure of this sensitivity.

Consider the linear system $Ax = b$, where A is nonsingular and b is nonzero. There is a unique solution x to this system which is also nonzero. Now suppose a small vector δb is added to b and consider a perturbed system $A\hat{x} = b + \delta b$. This system also has a unique solution $\hat{x}$, and let δx denote the difference between $\hat{x}$ and x, so that $\hat{x} = x + \delta x$.

Now it can be easily shown that

$$\frac{||\delta x||}{||x||} \leq ||A||||A^{-1}|| \frac{||\delta b||}{||b||} \tag{3.18}$$

Above equation provides a bound for $\frac{||\delta x||}{||x||}$ in terms of $\frac{||\delta b||}{||b||}$. The factor $||A||||A^{-1}||$ is called the *condition number* of A and is denoted as $\kappa(A)$.

$$\kappa(A) = ||A||||A^{-1}|| \tag{3.19}$$

If the coefficient matrix is symmetric and positive definite (that is, all it's eigenvalues are positive) above definition of condition number becomes

$$\kappa(A) = \frac{\lambda_{max}}{\lambda_{max}} \tag{3.20}$$

But in general (for any nonsingular coefficient matrix A) some of the eigenvalues will be complex and for that case, the following definition of condition number has been used.

$$\kappa(A) = \max_i |\lambda_i| / \min_i |\lambda_i| \tag{3.21}$$

Thus if $\kappa(A)$ is not too large, then small values of $||\delta b||/||b||$ imply small values of $||\delta x||/||x||$. That is, the system is not overly sensitive to perturbations in b. Thus if $\kappa(A)$ is not too large, one can say that A is *well conditioned.* If, on the other hand, $\kappa(A)$ is large, a small value of $||\delta b||/||b||$ does not guarantee that $||\delta x||/||x||$ will be small. Since, in this case, there definitely are choices of b and δb for which resulting $||\delta x||/||x||$ is much larger than the resulting $||\delta b||/||b||$. In other words, the system is potentially very sensitive to perturbations in b. Thus if $\kappa(A)$ is large, A is said to be *ill conditioned.*

The best possible condition number is 1. Of course, the condition number (definition given by Equation 3.19) depends on the definition of norm, and a given matrix have a large condition number with respect to one norm and small condition number with respect to another. As such, there is no cutoff between the ill-conditioned and well-conditioned matrices. Any such (fuzzy) boundary depends upon a number of factors, including the accuracy of data used, the precision of floating point numbers and the amount of error one is willing to tolerate in the computed solution.

3.3.3 Computation of Eigenvalues using MATLAB

Here, MATLAB function eig has been used for estimating eigenvalues. This function produces a vector e containing the eigenvalues of A when used in the following format.

$$e = \text{eig}(A)$$

For real matrices, eig(A) uses the EISPACK routines BALANC, BALBAK, ORTHES, and HQR2. BALANC and BALBAK balance the input matrix. ORTHES converts a real general matrix to Hessenberg by using orthogonal similarity transformations. Then, HQR2 finds the eigenvalues of a real upper Hessenberg matrix by the *QR algorithm.* Thus the whole process can be summarized in following three points.

1. Balancing of the input matrix

2. Reduction of the balanced matrix to upper Hessenberg form

3. Calculation of eigenvalues of upper Hessenberg matrix using *QR algorithm*

now these points can be further described as follows.

Balancing

The sensitivity of eigenvalues to rounding errors during the execution of the algorithm can be reduced by the procedure of *balancing*. The errors in the eigensystem found by the numerical procedure are generally proportional to the the Euclidean norm of the matrix, that is, to the square root of the sum of the squares of the elements. The idea of balancing is to use similarity transformations by diagonal matrices to make rows and columns of the matrix have comparable norms, thus reducing the overall norm of the matrix while leaving the eigenvalues unchanged. A symmetric matrix is already balanced.

The cost of balancing is n^2 flop counts, where n is the order of the input matrix and the time taken by the subroutine BALANC is never more than a few percent of the total time required to find the eigenvalues. It is therefore recommended that you always balance nonsymmetric matrices. It never hurts, and it can substantially improve the accuracy of the eigenvalues computed for a badly balanced matrix.

Reduction to Hessenberg form

To understand the need for reduction to Hessenberg form, first one need to learn something about the *QR algorithm*. The *QR algorithm* is an iterative process for finding eigenvalues. It is based on *QR decomposition*, which is a direct procedure related to Gram-Schmidt process. A single iteration of the *QR algorithm* is called a *QR step* or *QR iteration*. In a *QR step* for a general matrix, the cost of *QR factorization* is $\frac{4n^3}{3}$ and then the cost of matrix multiplication R_iQ_i is $2n^3$. Thus the total cost per *QR step* for a general matrix is $\frac{10n^3}{3}$. Reduction to upper Hessenberg form is equally expensive and requires $\frac{10n^3}{3}$, but it has to be done once, because upper Hessenberg form is preserved by the *QR algorithm*. Now since *QR iterations* with the upper Hessenberg matrix are relatively inexpensive and each can be done costing only $O(n^2)$ flops, *QR algorithm* is applied only after converting the input matrix to Hessenberg form.

Algorithm for Householder reduction to Hessenberg form is as follows (Watkins [2002]).

Algorithm 3.1: Householder Reduction to Hessenberg form

for $k = 1$ to $n - 2$

$x = A_{k+1:m,k}$

$v_k = \text{sign}(x)||x||_2 e_1 + x$

$v_k = v_k/||v_k||_2$

$A_{k+1:m,k:m} = A_{k+1:m,k:m} - 2v_k(v_k^T A_{k+1:m,k:m})$

$A_{1:m,k+1:m} = A_{1:m,k+1:m} - 2(A_{1:m,k+1:m}v_k)v_k^T$

end

Here sign is a function such that sign(x) returns an array y of the same size as x, where each element of y is:
1 if the corresponding element of x is greater than zero
0 if the corresponding element of x equals zero
-1 if the corresponding element of x is less than zero
and $e_1 = (1, 0, ..., 0)$.

QR algorithm

As already described, QR algorithm is an iterative process for finding eigenvalues. It is based on QR decomposition, which is a direct procedure related to Gram-Schmidt process. During iterations Hessenberg form of the matrix is preserved and the diagonal elements finally converge to eigenvalues. QR algorithm can be summarized as follows (Watkins [2002]).

Algorithm 3.2: QR algorithm

If A^0 is the Hessenberg reduction of A, then
Set $A = (Q^0)^T A^0 Q^0$
for $k = 1, 2...$

Pick a shift μ^k	e.g., choose $\mu^k = A_{m,m}^{k-1}$
$Q^k R^k = A^{k-1} - \mu^k I$	QR factorization of $A^{k-1} - \mu^k I$
$A_k = R^k Q^k + \mu^k I$	Recombine factors in reverse order

end

3.3.4 Test cases for the Computation of Eigenvalues and Condition Numbers

For the computation of eigenvalues and condition numbers, following two cases of the advection-diffusion problem have been considered.

Unsteady Nonlinear Advection-Diffusion

Governing equation for this case is given as

$$\frac{\partial\theta}{\partial\tau}+\frac{\partial(u^*\theta)}{\partial X}=\frac{1}{\mathrm{Pe}}\left[\frac{\partial}{\partial X}\left(\tilde{k}\frac{\partial\theta}{\partial X}\right)+\frac{\partial}{\partial Y}\left(\tilde{k}\frac{\partial\theta}{\partial Y}\right)\right] \tag{3.22}$$

u^* is assumed to be a parabolic flow profile given by

$$u^*=\frac{3}{2}\left[1-Y^2\right] \tag{3.23}$$

and

$$\tilde{k}=1-(0.4)\theta \tag{3.24}$$

Steady Linear Advection-Diffusion

Governing equation for the steady and linear is a special case of Equation 3.22.

$$\frac{\partial(u^*\theta)}{\partial X}=\frac{1}{\mathrm{Pe}}\left(\frac{\partial^2\theta}{\partial X^2}+\frac{\partial^2\theta}{\partial Y^2}\right) \tag{3.25}$$

Discretization Formulation

Governing Equations 3.22 and 3.25 has been discretized in the same way as in Section 3.2.3. Moreover, the initial and boundary conditions used are also exactly the same as given in Section 3.2.2.

This gives the following discretized equation for the unsteady-nonlinear case.

$$
\begin{aligned}
&\left(3u_P^*\mathrm{Pe}_c\right)\theta_{WW}^{p+1} - \left(19u_P^*\mathrm{Pe}_c + 6\left(\tilde{k}_W + 4\tilde{k}_P - \tilde{k}_E\right)\right)\theta_W^{p+1} \\
&+ \left[r + 7u_P^*\mathrm{Pe}_c + 48\tilde{k}_P\left(1+R^2\right)\right]\theta_P^{p+1} - \left(6\left(\tilde{k}_E + 4\tilde{k}_P - \tilde{k}_W\right) - 9u_P^*\mathrm{Pe}_c\right)\theta_E^{p+1} \\
&- \left(6R^2\left(\tilde{k}_N + 4\tilde{k}_P - \tilde{k}_S\right) - u_N^*\mathrm{Pe}_c\right)\theta_N^{p+1} - \left(6R^2\left(\tilde{k}_S + 4\tilde{k}_P - \tilde{k}_N\right) - u_S^*\mathrm{Pe}_c\right)\theta_S^{p+1} \\
&- \left(u_N^*\mathrm{Pe}_c\right)\theta_{NW}^{p+1} - \left(u_S^*\mathrm{Pe}_c\right)\theta_{SW}^{p+1} = r\theta_P^p
\end{aligned} \tag{3.26}
$$

where Pe_c is cell Peclet number, defined as $\mathrm{Pe}_c = \mathrm{Pe}(\Delta X)$, $R = (\Delta X)/(\Delta Y)$ and

$$r = \frac{24(\Delta X)\mathrm{Pe}_c}{\Delta\tau}.$$

For the steady-linear case it becomes.

$$
\begin{aligned}
&\left(3u_P^*\mathrm{Pe}_c\right)\theta_{WW} - \left(19u_P^*\mathrm{Pe}_c + 24\right)\theta_W + \left[7u_P^*\mathrm{Pe}_c + 48\left(1+R^2\right)\right]\theta_P \\
&- \left(24 - 9u_P^*\mathrm{Pe}_c\right)\theta_E - \left(24R^2 - u_N^*\mathrm{Pe}_c\right)\theta_N - \left(24R^2 - u_S^*\mathrm{Pe}_c\right)\theta_S \\
&- \left(u_N^*\mathrm{Pe}_c\right)\theta_{NW} - \left(u_S^*\mathrm{Pe}_c\right)\theta_{SW} = 0
\end{aligned} \tag{3.27}
$$

Procedure for Calculation

For the steady-linear case the coefficient matrix is generated using the governing discretized Equation 3.27. In the unsteady-nonlinear case the very first matrix arising just after 10 time steps is considered where the time step is taken as 0.005. In both the cases, length of the computational domain is taken 50 and the original coefficient matrix has the structure shown below in Figure 3.9.
Then in both the cases preconditioned matrices are obtained from the left-right incomplete lower-upper (ILU) preconditioning of the original coefficient matrix. Preconditioned matrix is given by

$$P = L^{-1}AU^{-1} \tag{3.28}$$

Where L and U are the lower and upper triangular matrices obtained from LU decomposition of the original coefficient matrix A. LU decomposition is incomplete in the sense that complete band of the original coefficient matrix is not used. Infact, all the possible bands (semi-bandwidths shown in Figure 3.9 by arrows) are have been tested for this type of incomplete LU decomposition.

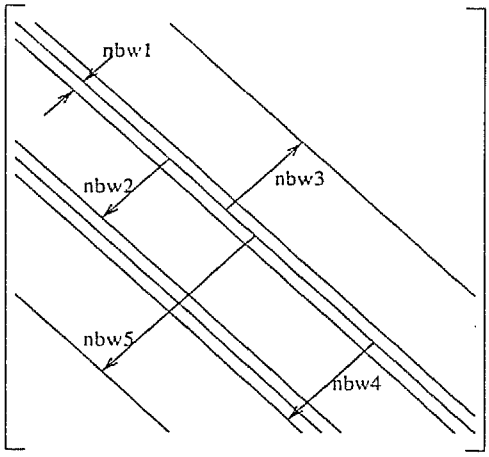


Figure 3.9: Original coefficient matrix

Finally, these matrices are supplied to MATLAB and it calculates all the eigenvalues as discussed in Section 3.3.3.

3.3.5 Results and Discussion

Following results have been presented for this analysis:

1. Variation of condition number with Peclet number for three different grid sizes, in the steady-linear case (Figure 3.10).

2. Variation of condition number with Peclet number for three different grid sizes, for the unsteady-nonlinear case (Figure 3.11).

3. Eigenvalue spectra of the original matrices in steady-linear problem for different values of Peclet number (Figure 3.12).

4. Eigenvalue spectra of the original matrices in unsteady-nonlinear problem for different values of Peclet number (Figure 3.13).

5. Eigenvalue spectra along with condition numbers for the original and pre-conditioned matrices, for the steady-linear case (Figures 3.14-3.28).

6. Eigenvalue spectra along with condition numbers for the original and pre-conditioned matrices, for the unsteady-nonlinear case (Figures 3.29-3.43).

Eigenvalue spectra shown in Figures 3.12 and 3.13 are for a 101×11 grid. All the other eigenvalue spectra, listed above in 5 and 6, have been presented for

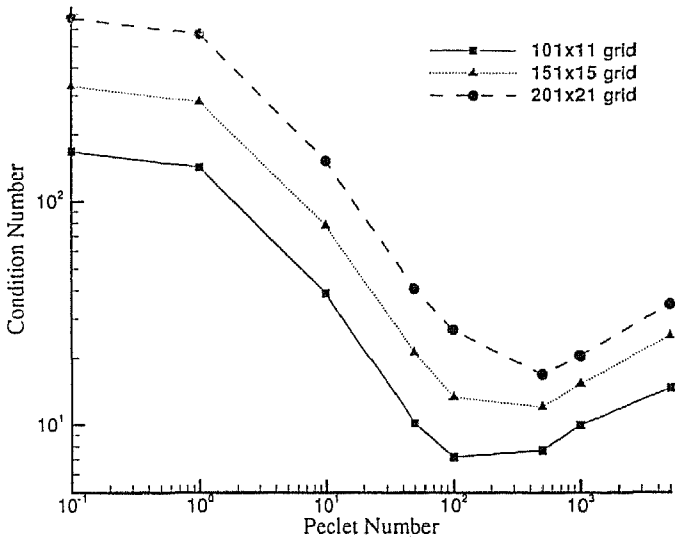


Figure 3.10: Variation of condition number with Peclet number for three different grid sizes in steady-linear case

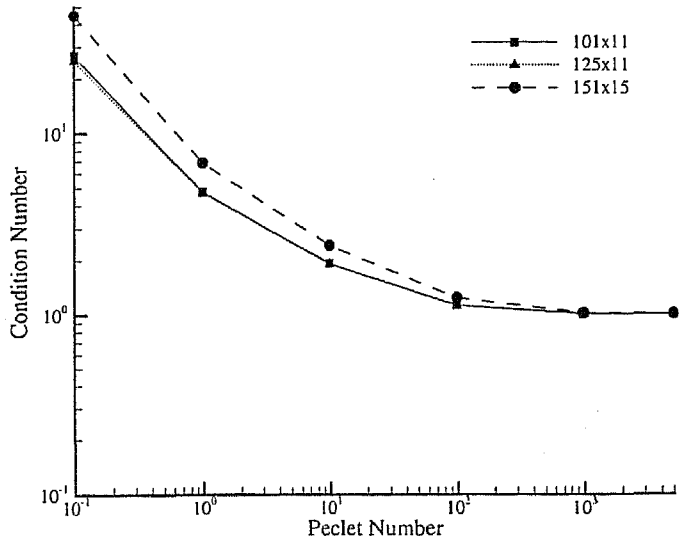


Figure 3.11: Variation of condition number with Peclet number for three different grid sizes in unsteady-nonlinear case

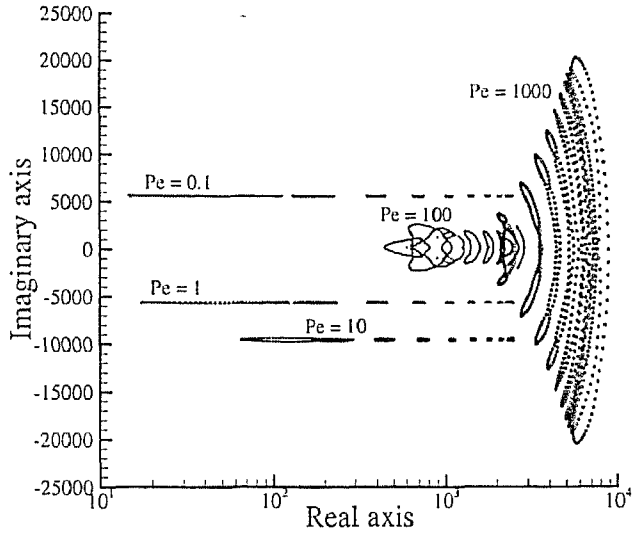


Figure 3.12: Eigenvalue spectra of the original matrices in steady-linear problem for different values of Peclet number.

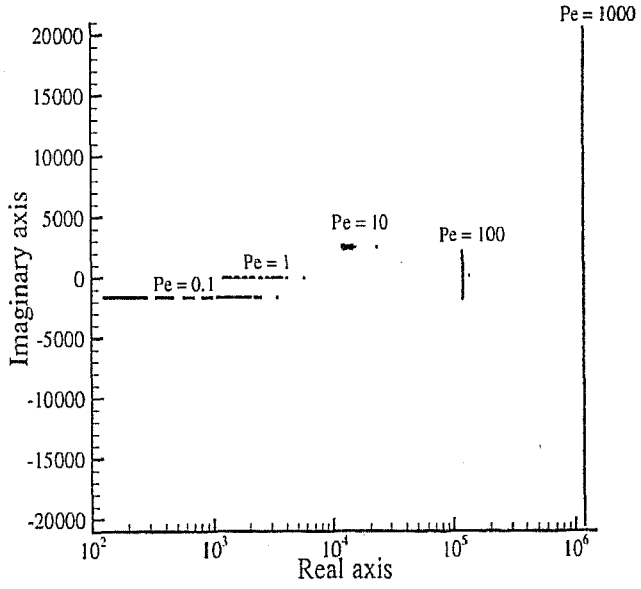


Figure 3.13: Eigenvalue spectra of the original matrices in unsteady-nonlinear problem for different values of Peclet number.

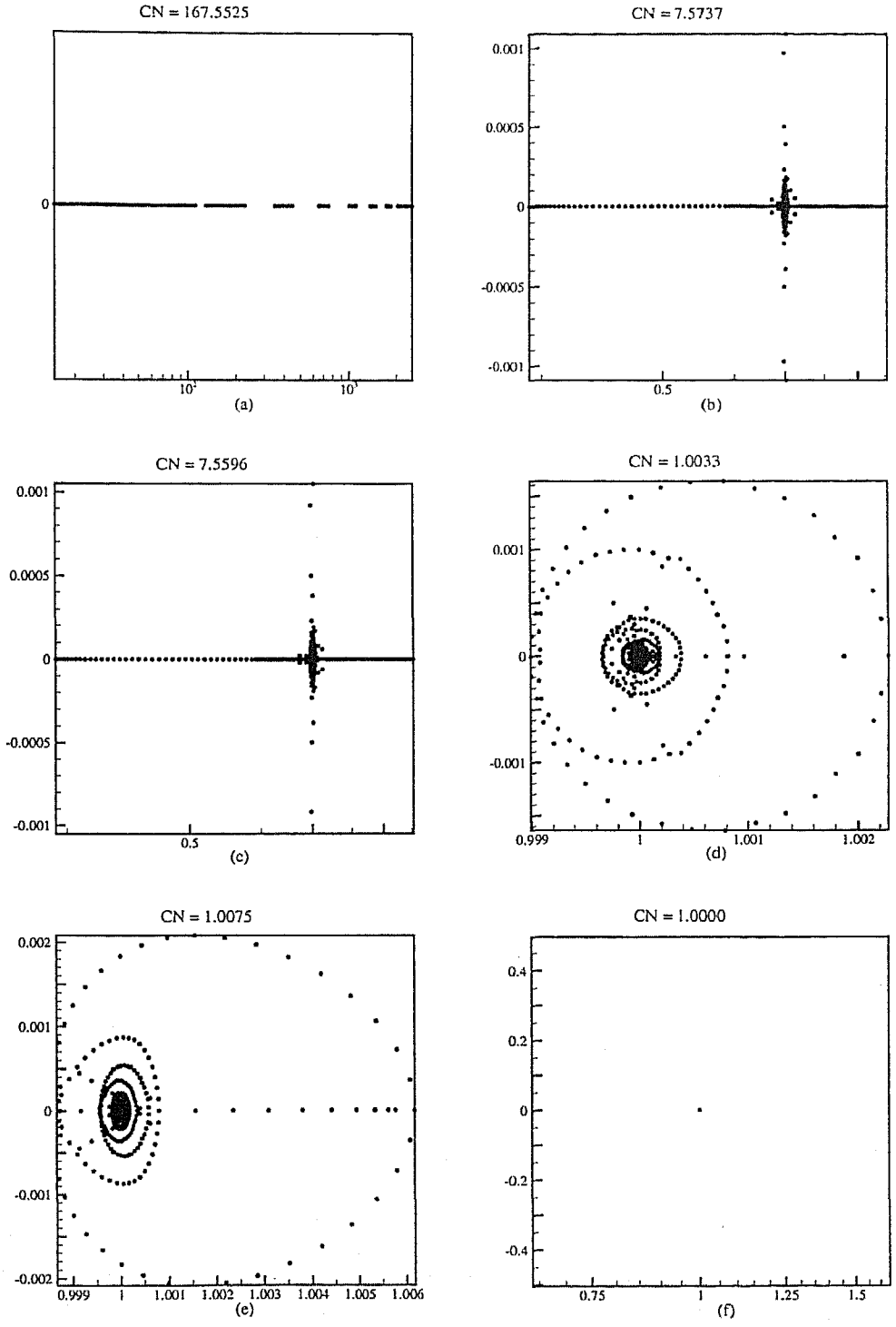


Figure 3.14: Eigenvalue spectrum and condition numbers for Pe = 0.10, using a 101×11 grid in the steady-linear case.

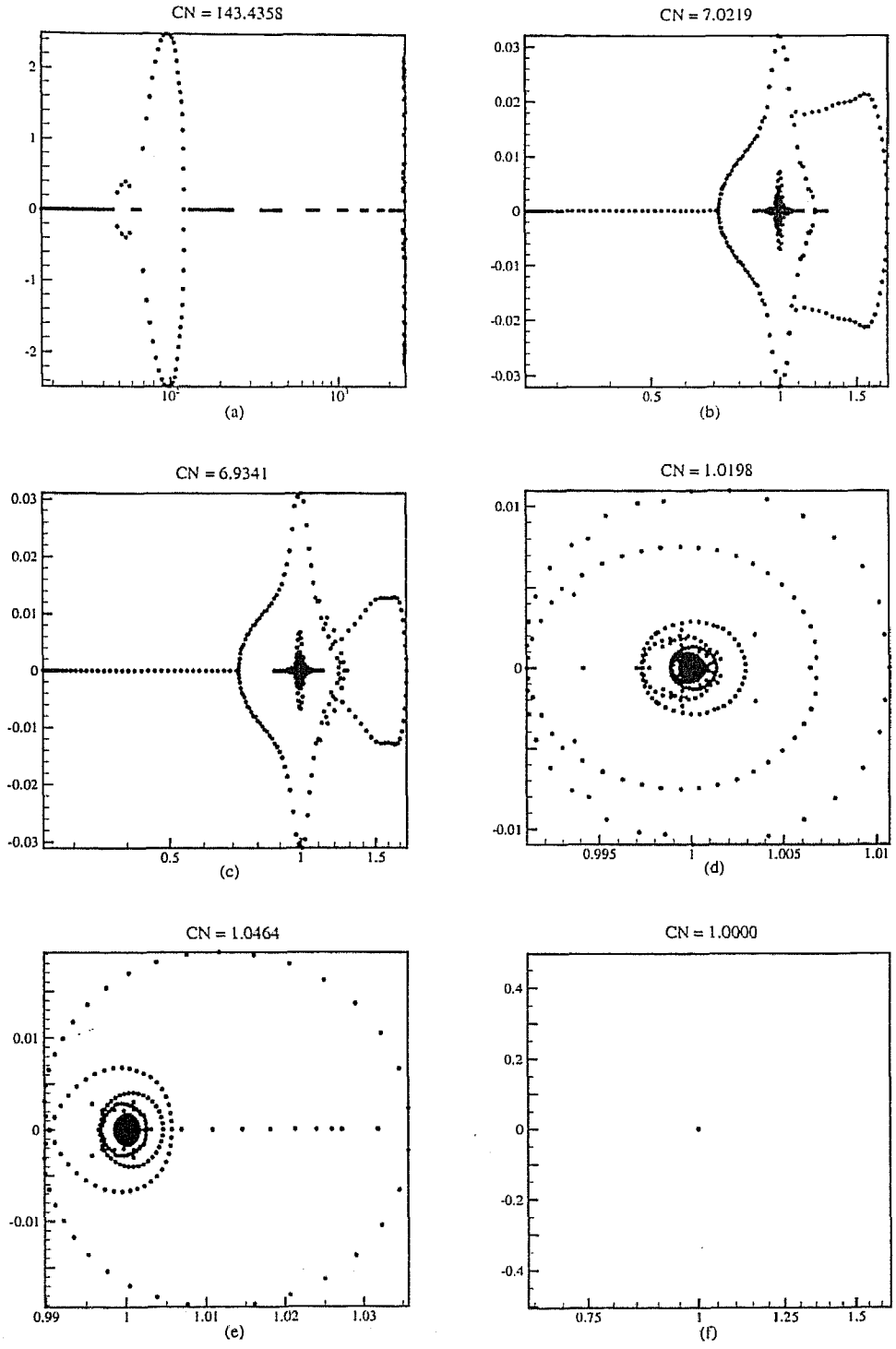


Figure 3.15: Eigenvalue spectrum and condition numbers for Pe = 1, using a 101 × 11 grid in the steady-linear case.

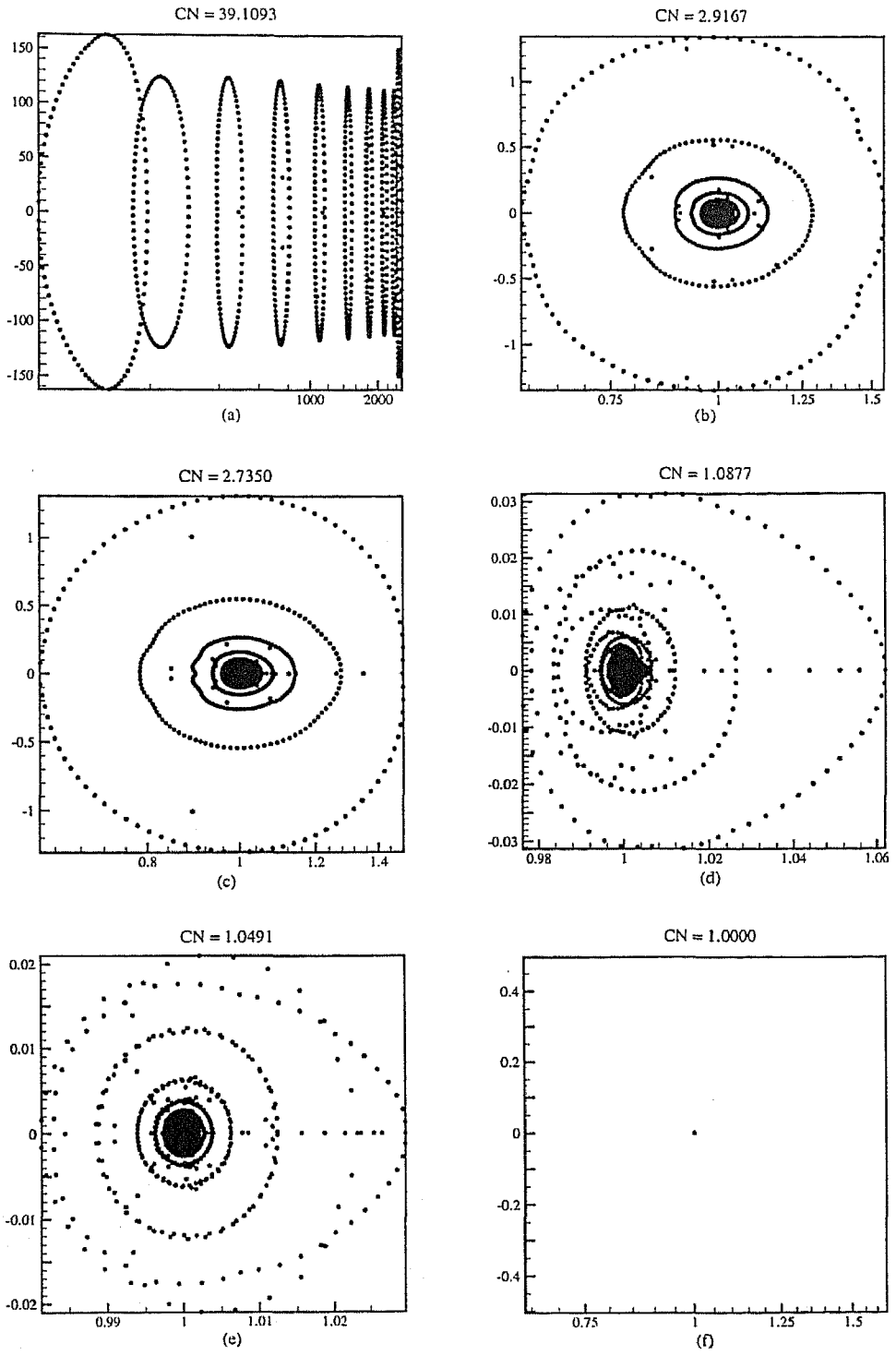


Figure 3.16: Eigenvalue spectrum and condition numbers for Pe = 10, using a 101×11 grid in the steady-linear case.

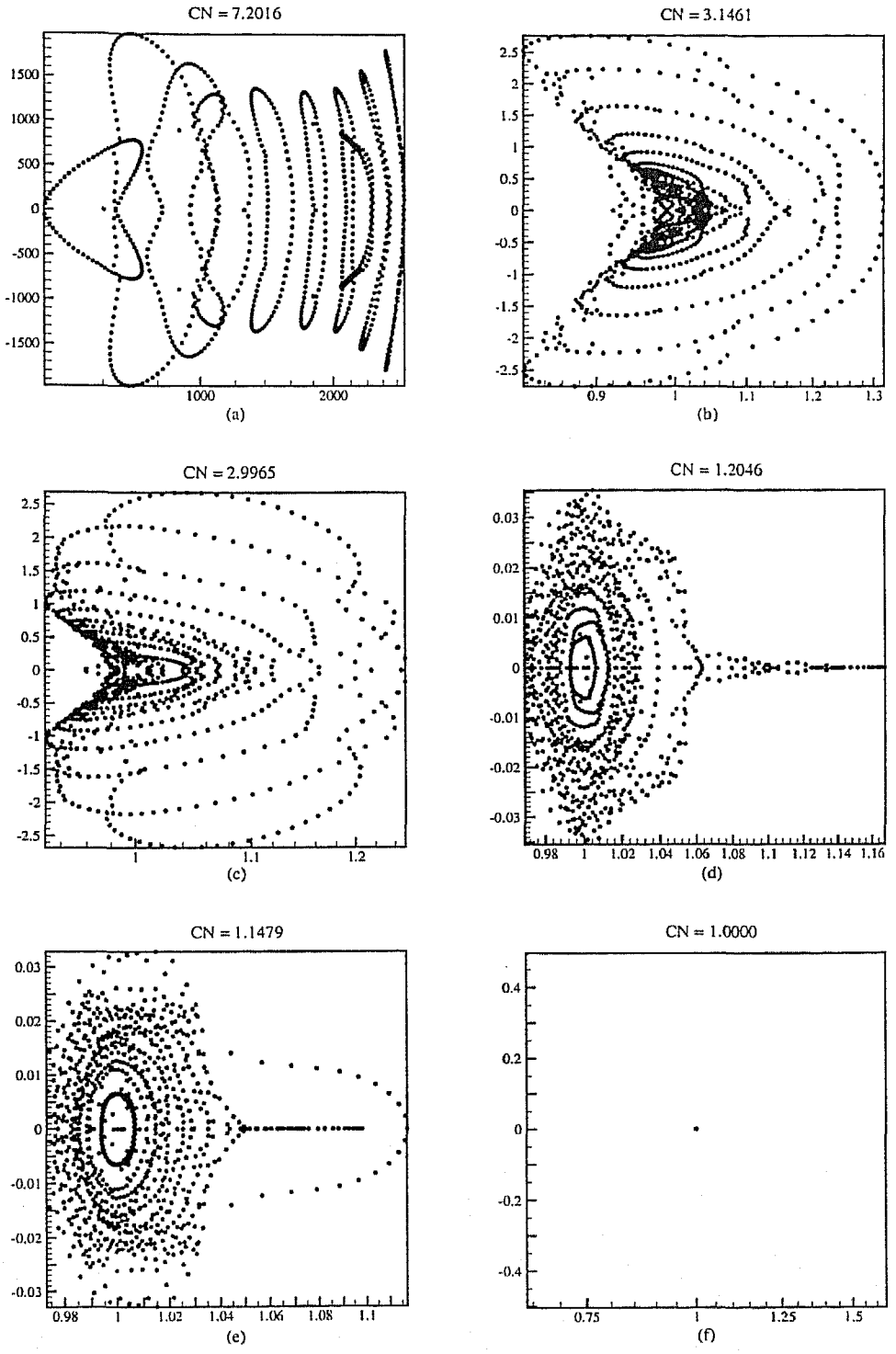


Figure 3.17: Eigenvalue spectrum and condition numbers for Pe = 100, using a 101 × 11 grid in the steady-linear case.

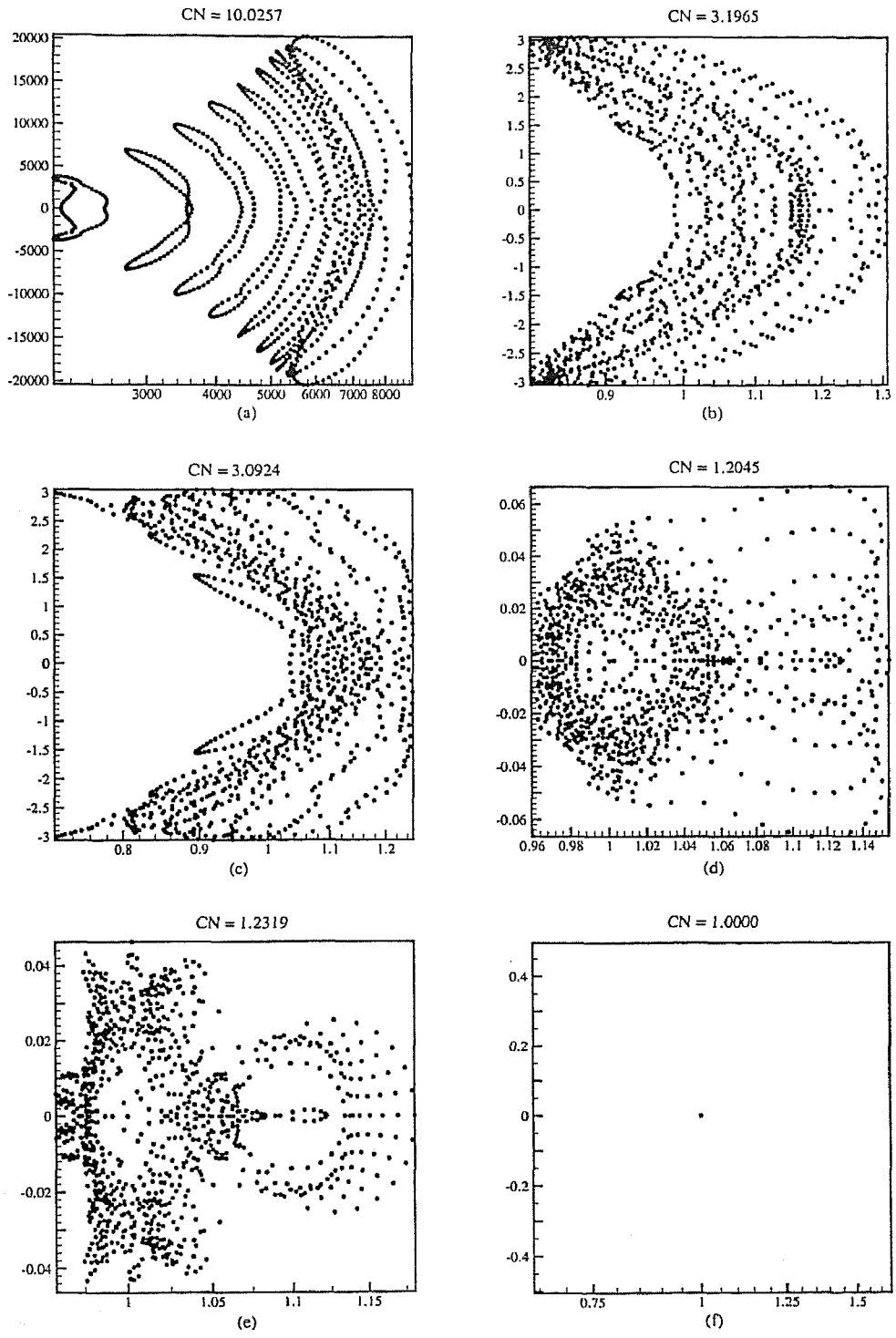


Figure 3.18: Eigenvalue spectrum and condition numbers for Pe = 1000, using a 101×11 grid in the steady-linear case.

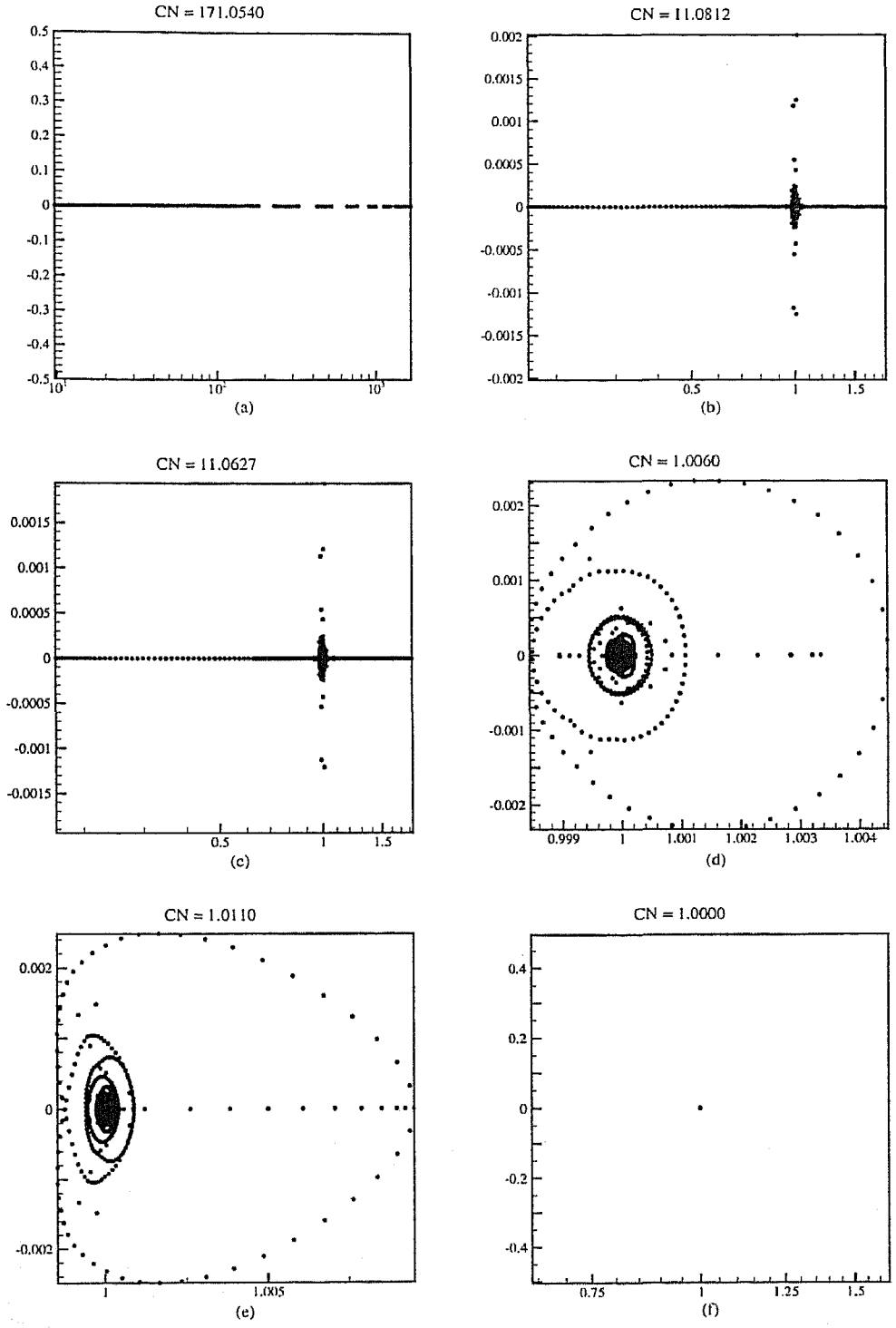


Figure 3.19: Eigenvalue spectrum and condition numbers for Pe = 0.10, using a 125 × 11 grid in the steady-linear case.

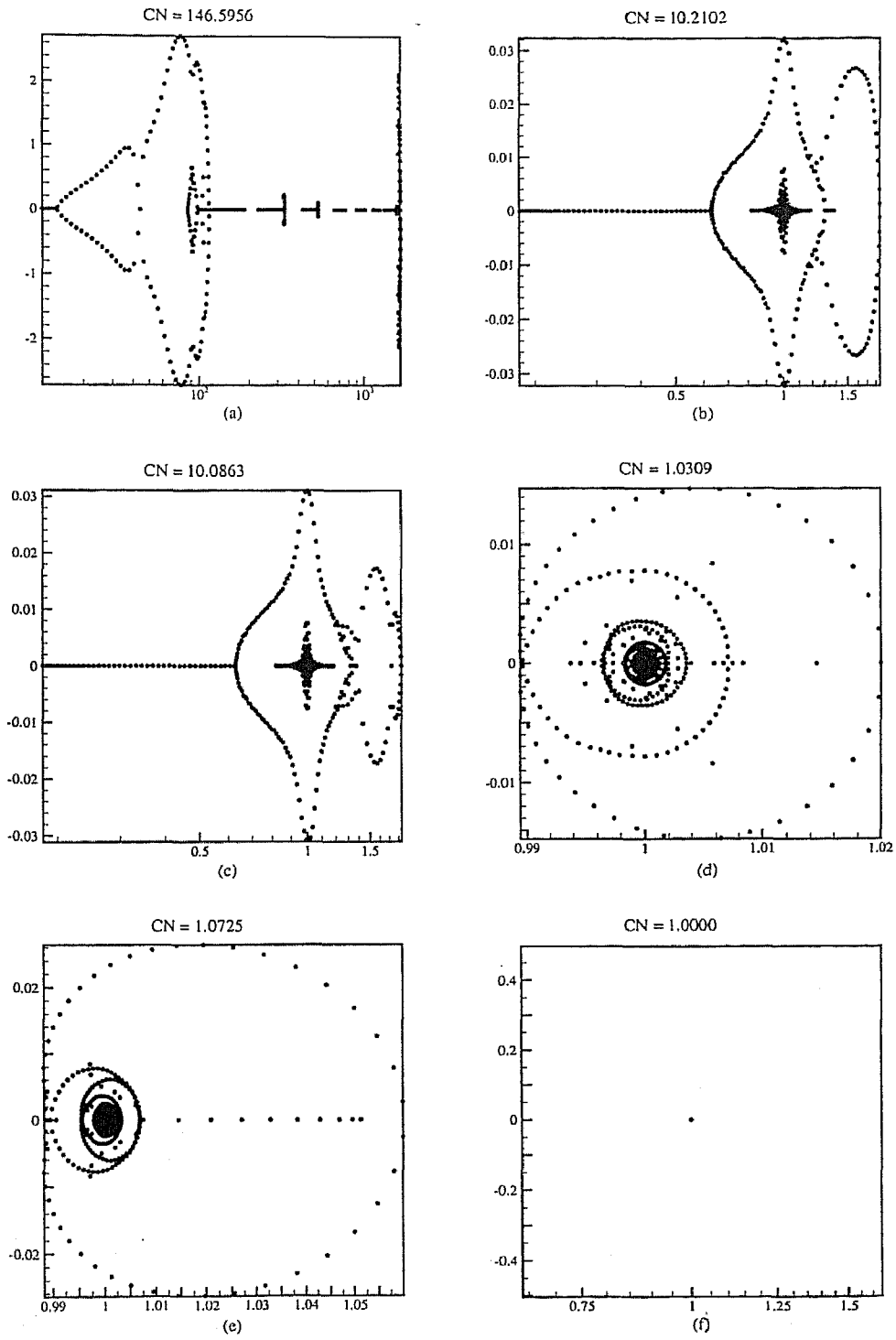


Figure 3.20: Eigenvalue spectrum and condition numbers for Pe = 1, using a 125 × 11 grid in the steady-linear case.

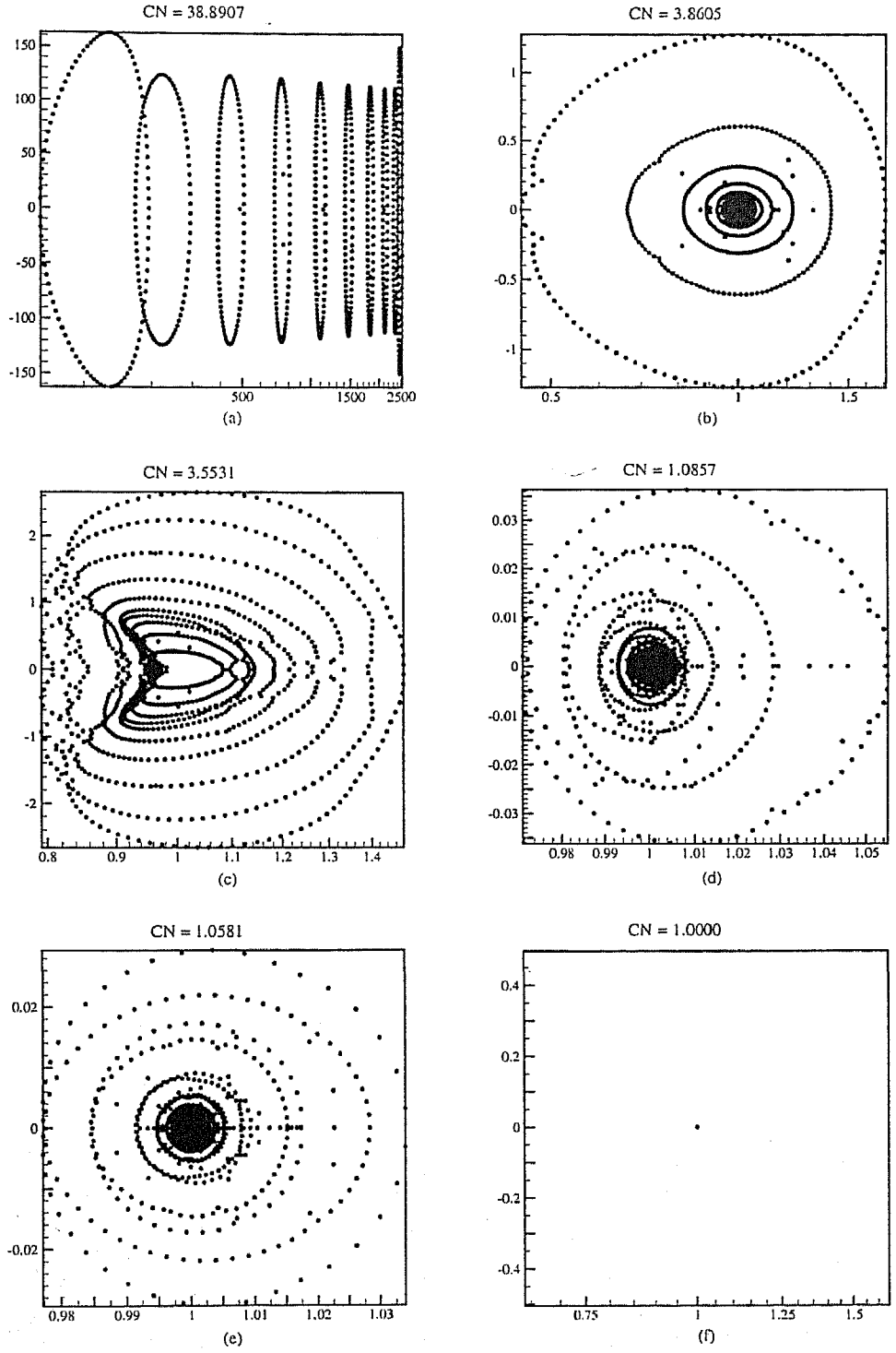


Figure 3.21: Eigenvalue spectrum and condition numbers for Pe = 10, using a 125 × 11 grid in the steady-linear case.

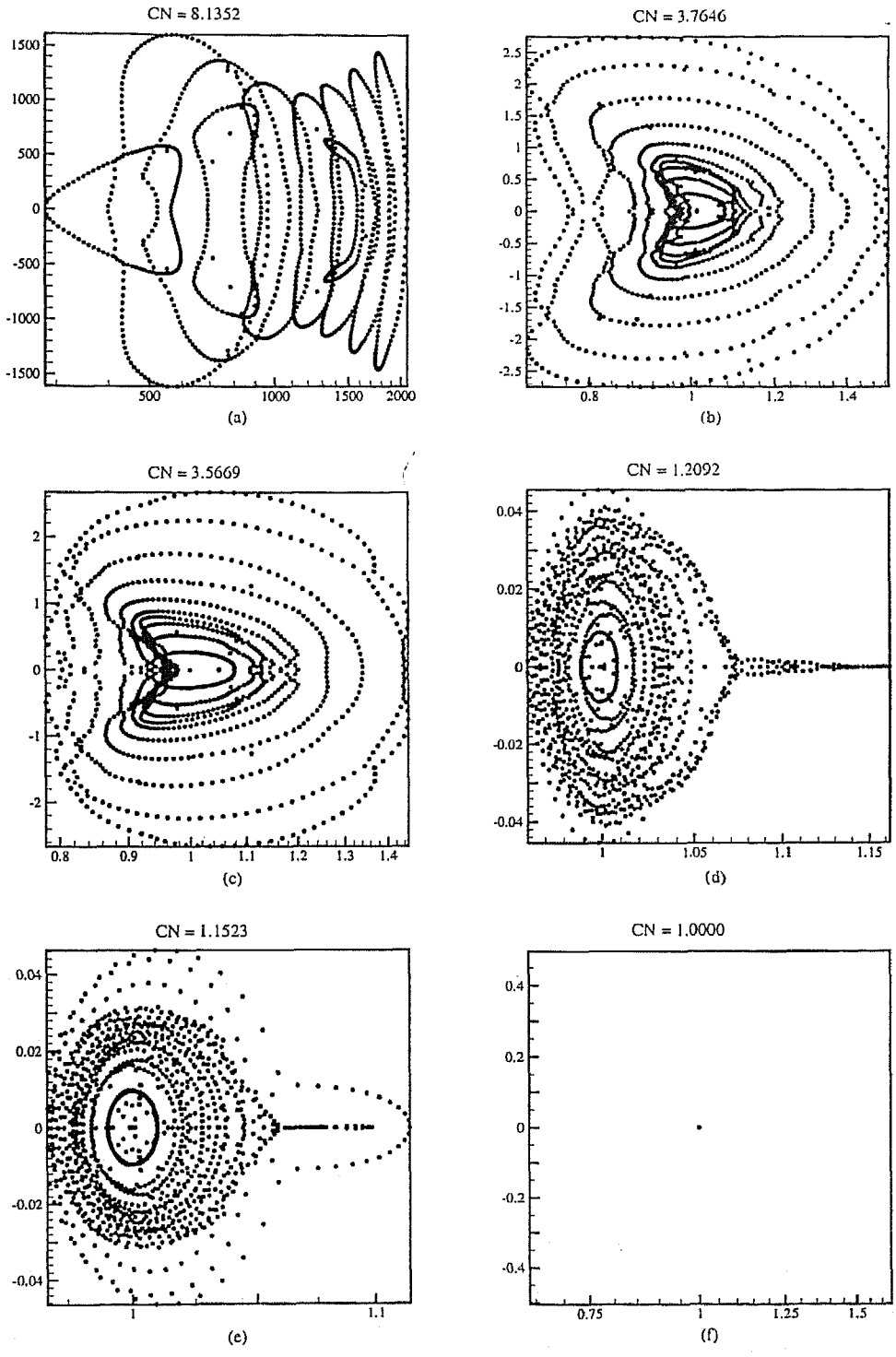


Figure 3.22: Eigenvalue spectrum and condition numbers for Pe = 100, using a 125×11 grid in the steady-linear case.

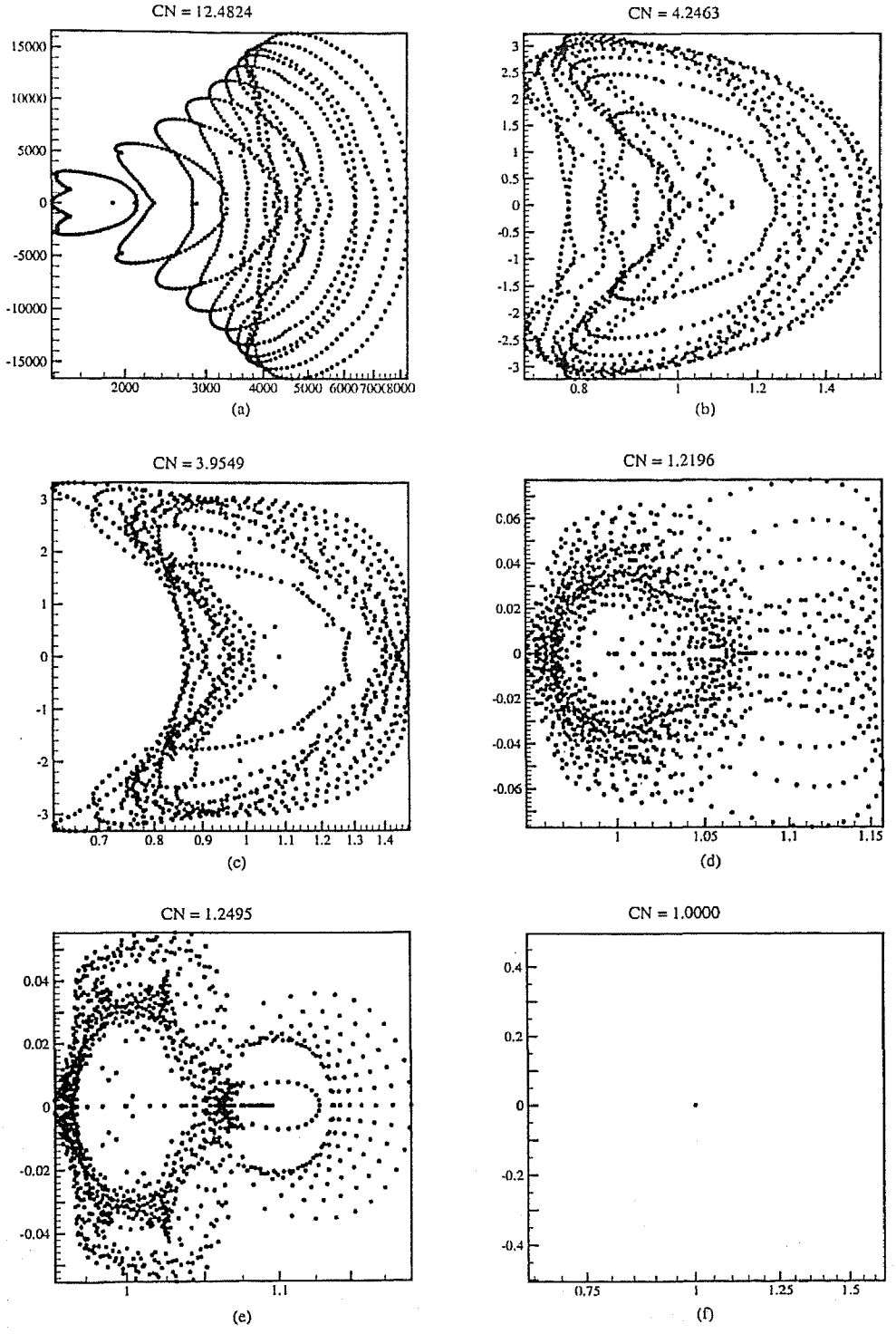


Figure 3.23: Eigenvalue spectrum and condition numbers for Pe = 1000, using a 125×11 grid in the steady-linear case.

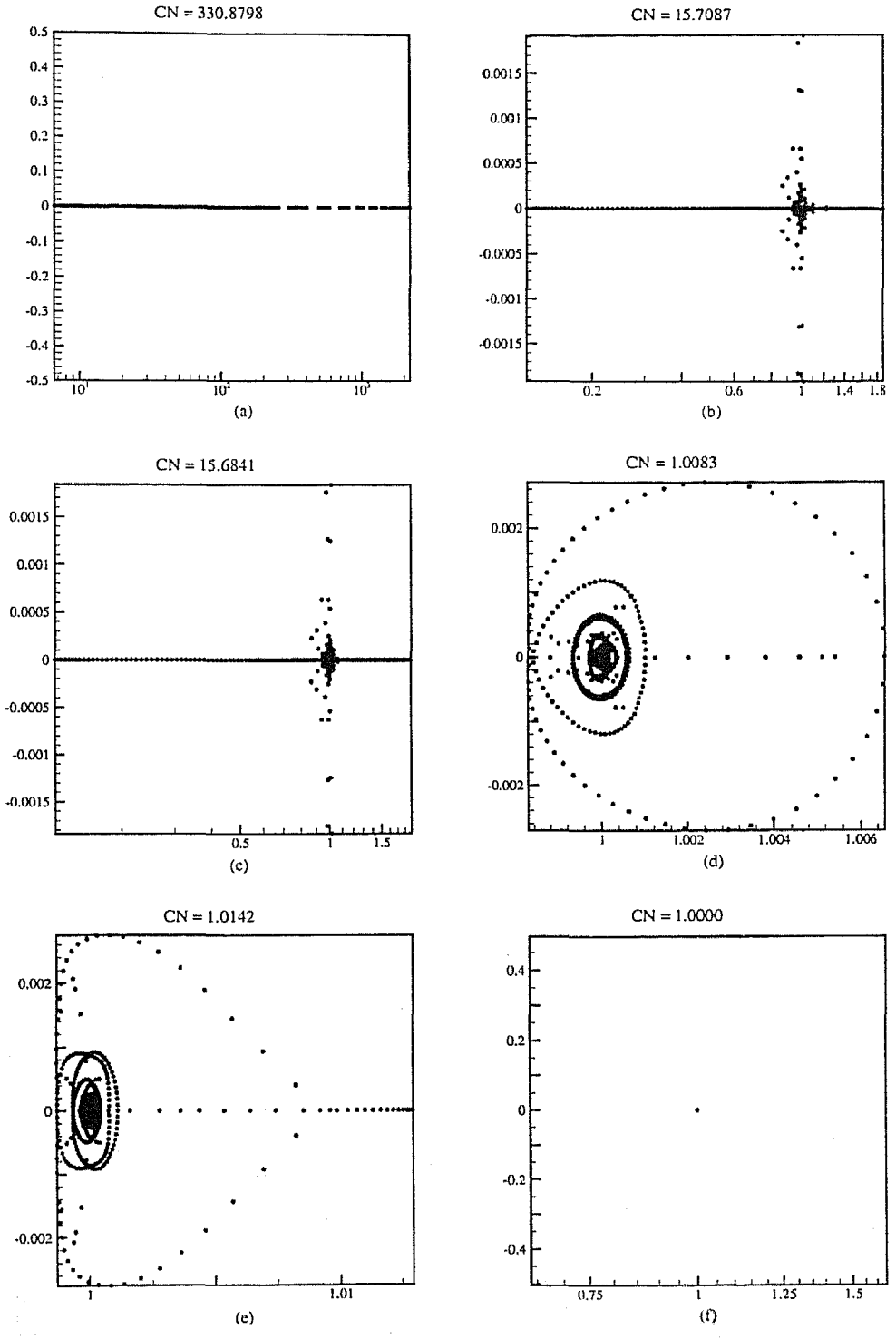


Figure 3.24: Eigenvalue spectrum and condition numbers for Pe = 0.10, using a 151 × 15 grid in the steady-linear case.

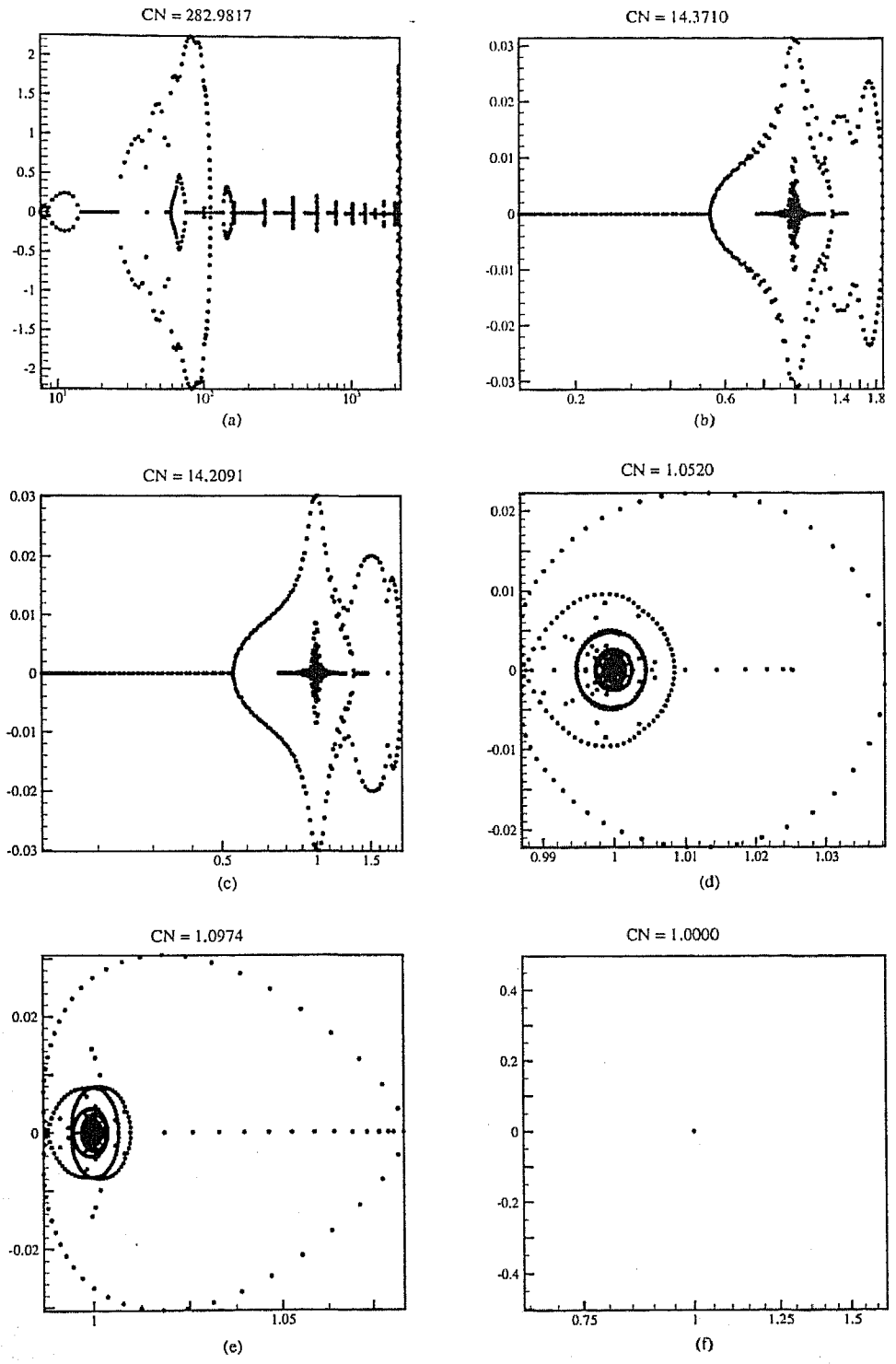


Figure 3.25: Eigenvalue spectrum and condition numbers for Pe = 1, using a 151 × 15 grid in the steady-linear case.

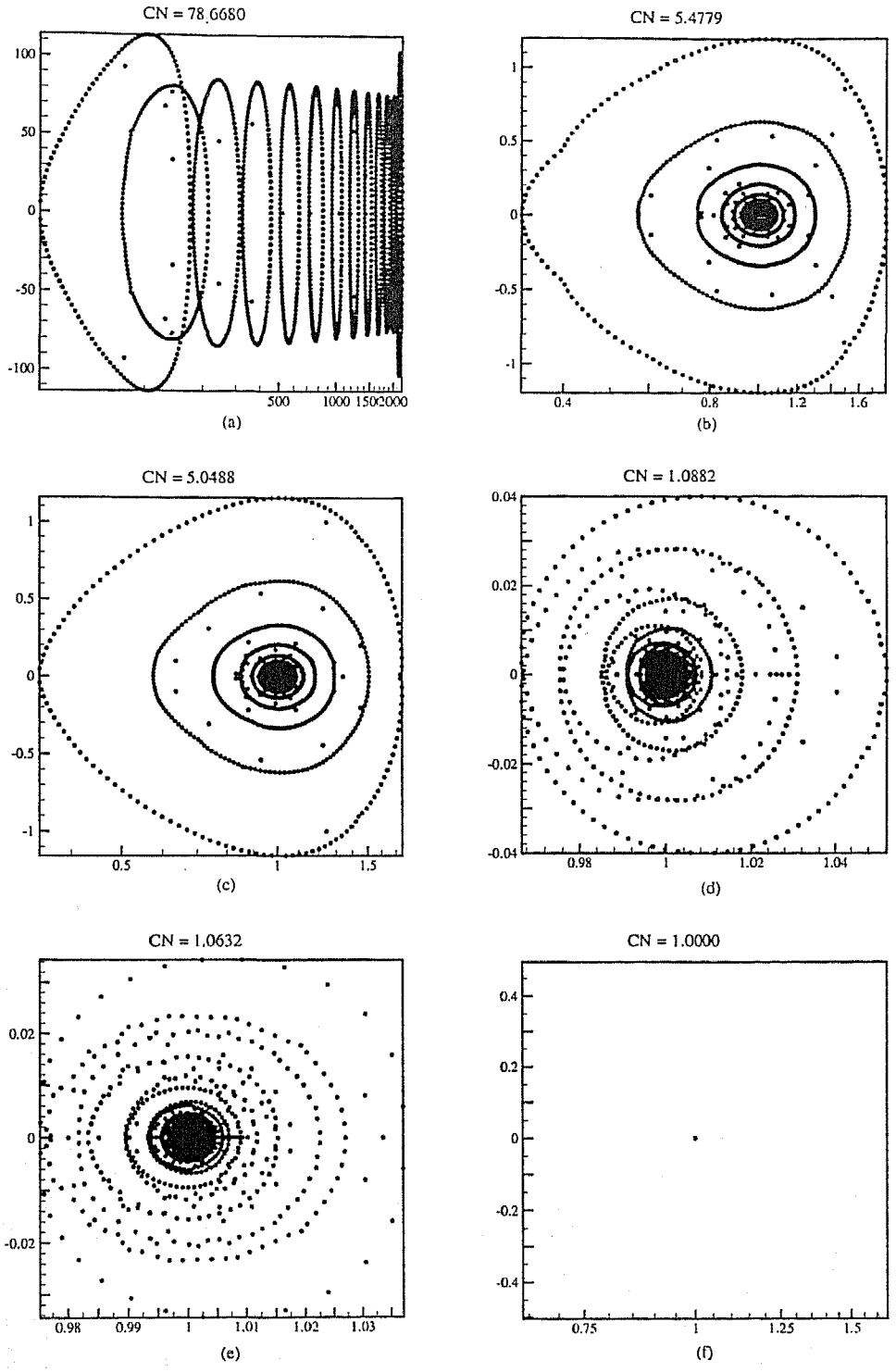


Figure 3.26: Eigenvalue spectrum and condition numbers for Pe = 10, using a 151×15 grid in the steady-linear case.

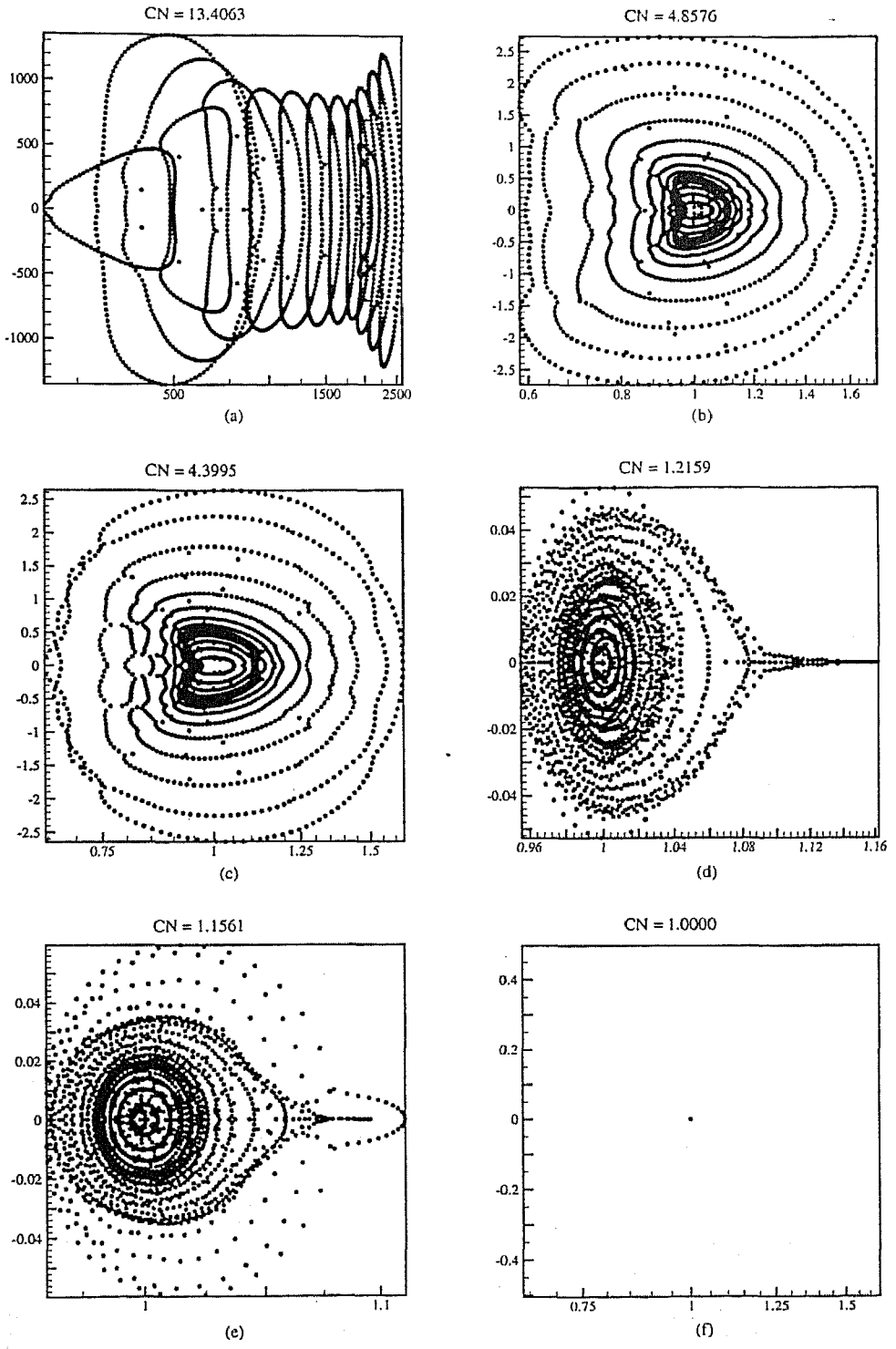


Figure 3.27: Eigenvalue spectrum and condition numbers for Pe = 100, using a 151×15 grid in the steady-linear case.

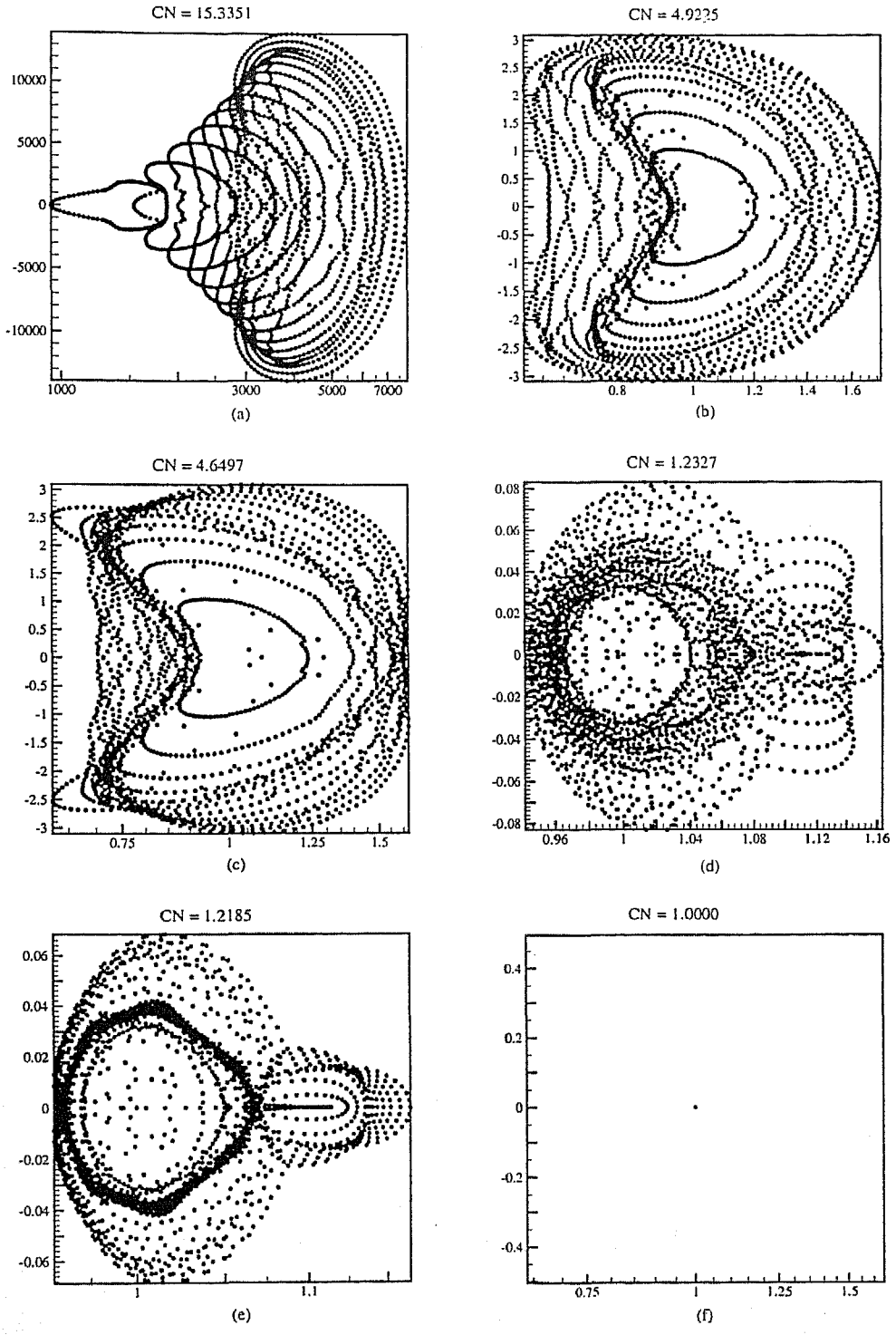


Figure 3.28: Eigenvalue spectrum and condition numbers for Pe = 1000, using a 151 × 15 grid in the steady-linear case.

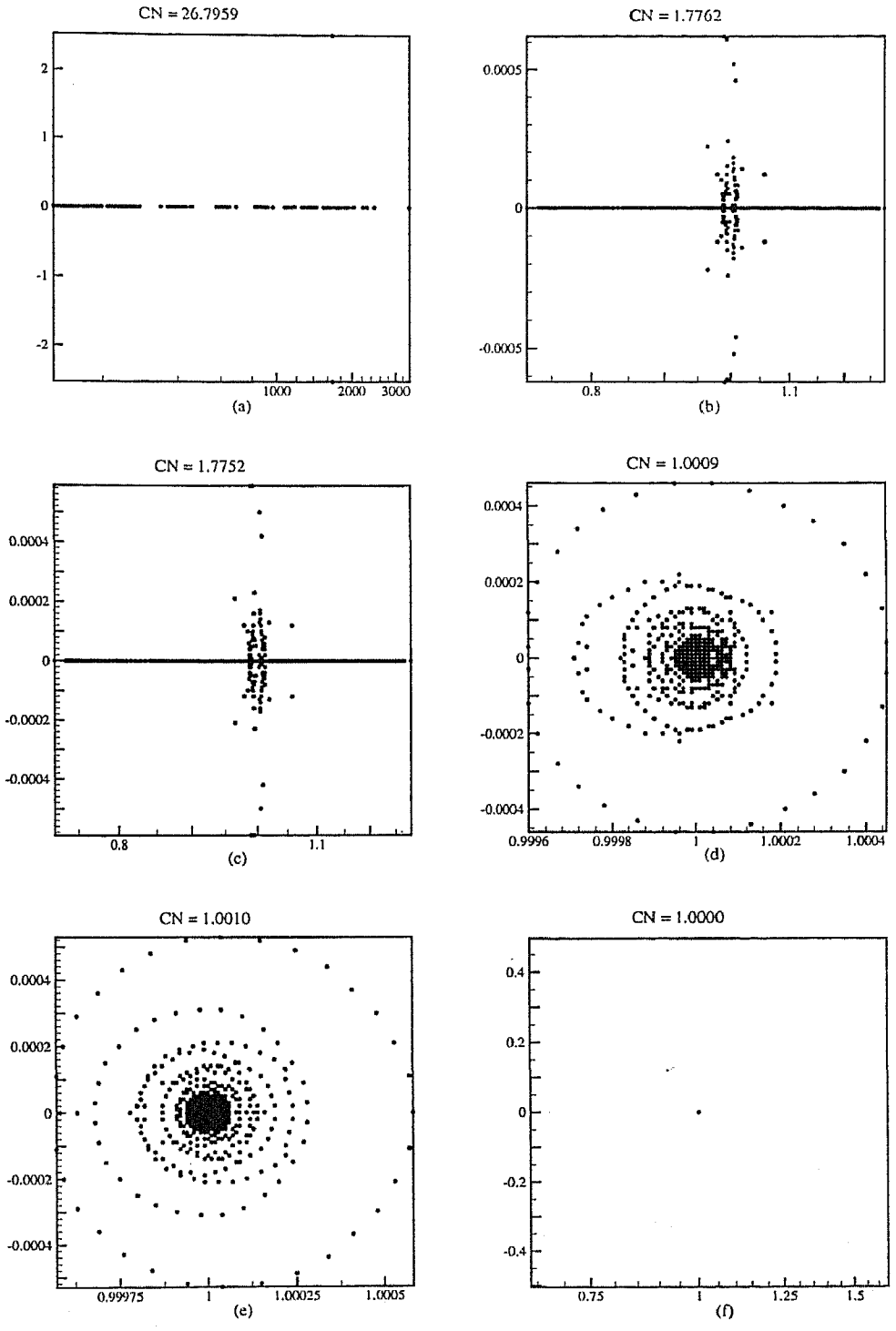


Figure 3.29: Eigenvalue spectrum and condition numbers for Pe = 0.10, using a 101×11 grid in the unsteady-nonlinear case.

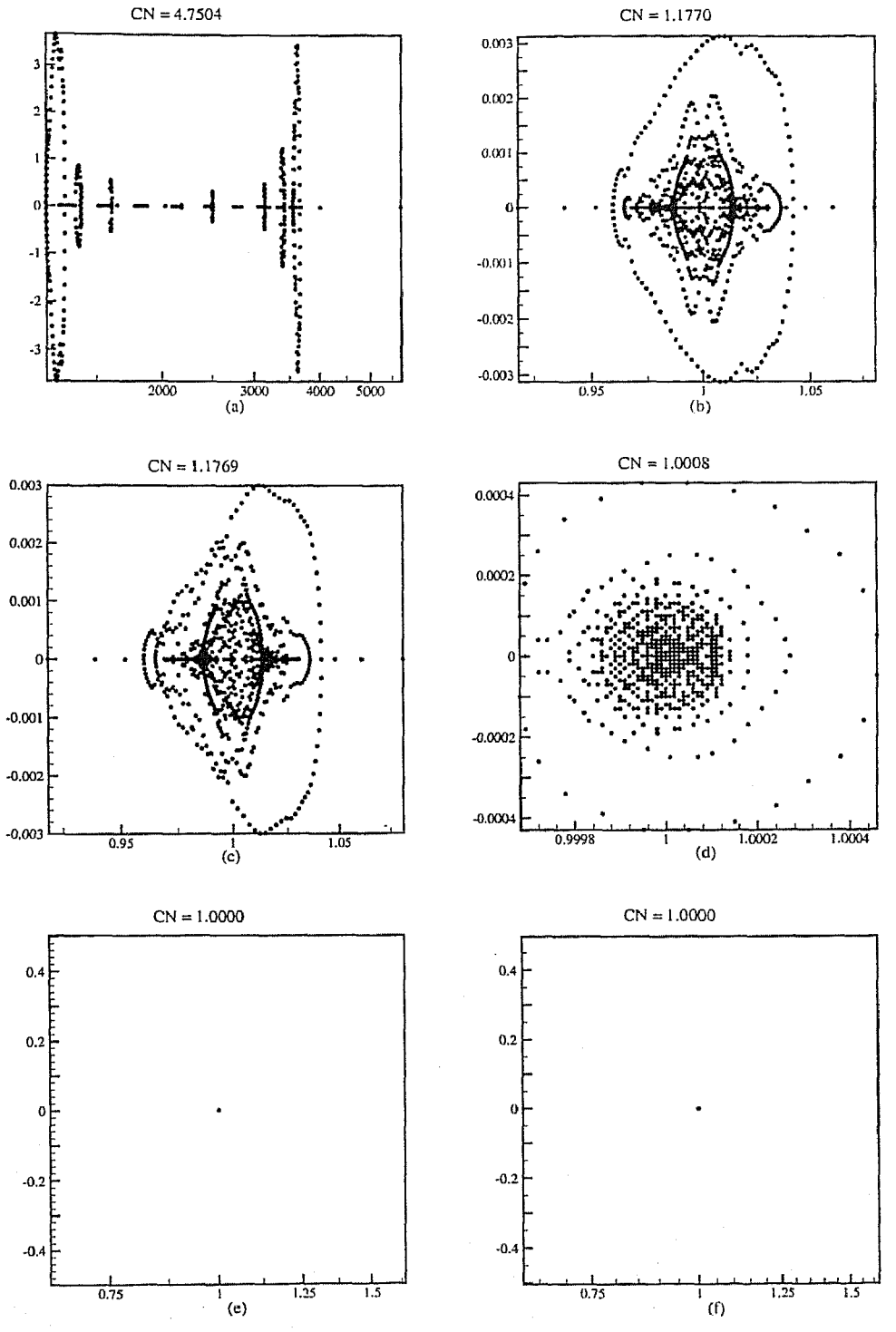


Figure 3.30: Eigenvalue spectrum and condition numbers for Pe = 1, using a 101 × 11 grid in the unsteady-nonlinear case.

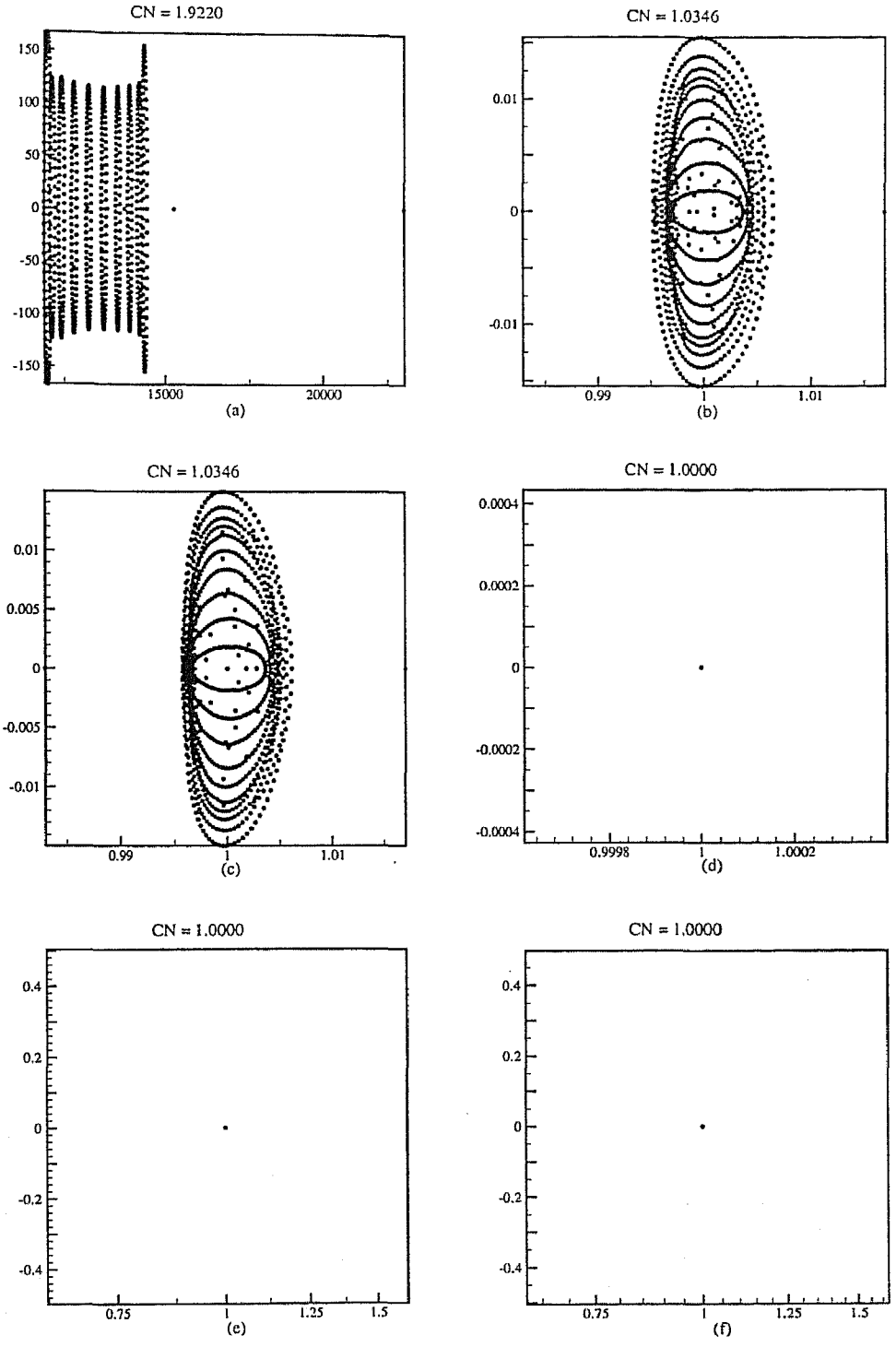


Figure 3.31: Eigenvalue spectrum and condition numbers for Pe = 10, using a 101×11 grid in the unsteady-nonlinear case.

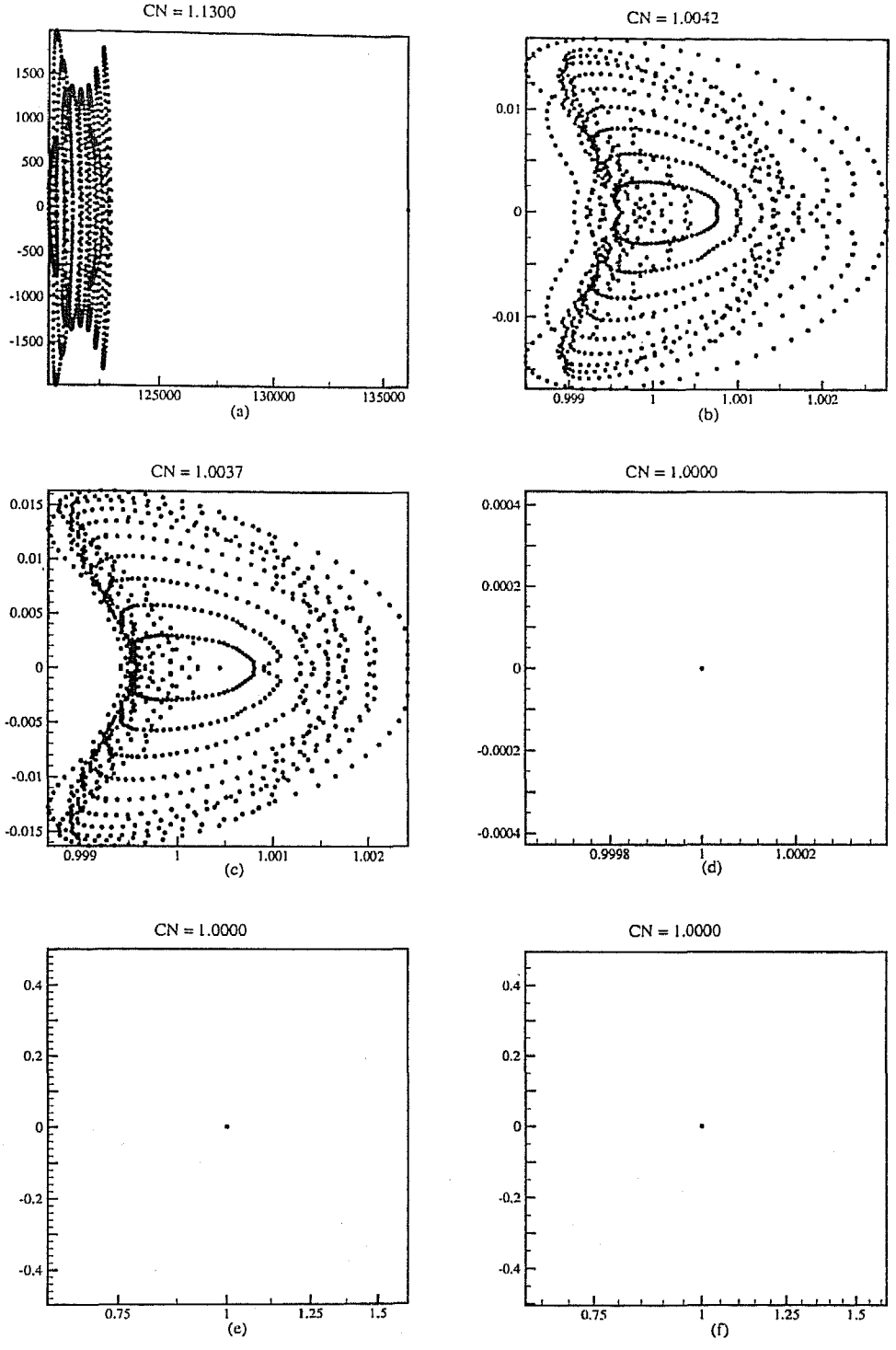


Figure 3.32: Eigenvalue spectrum and condition numbers for Pe = 100, using a 101×11 grid in the unsteady-nonlinear case.

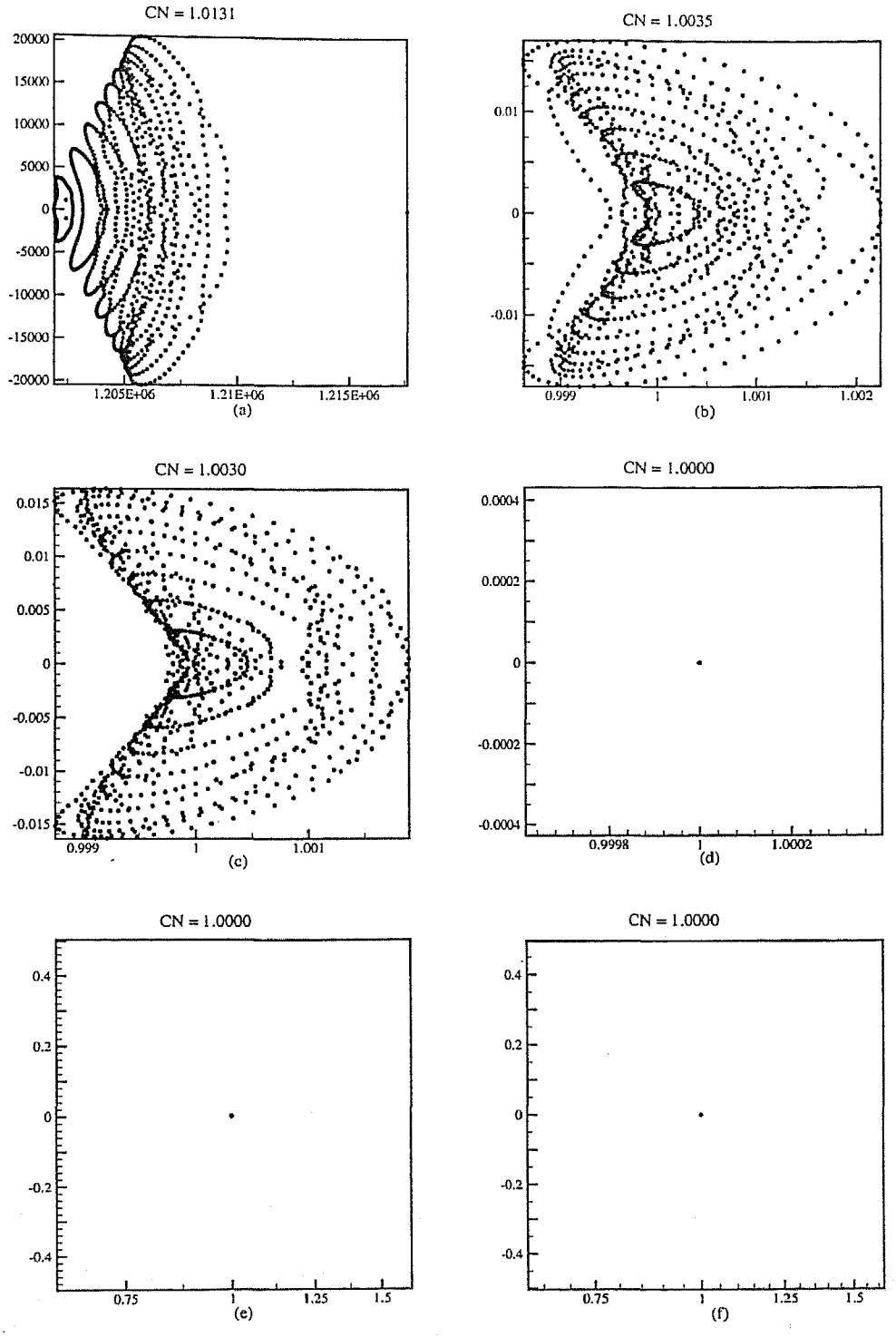


Figure 3.33: Eigenvalue spectrum and condition numbers for Pe = 1000, using a 101×11 grid in the unsteady-nonlinear case.

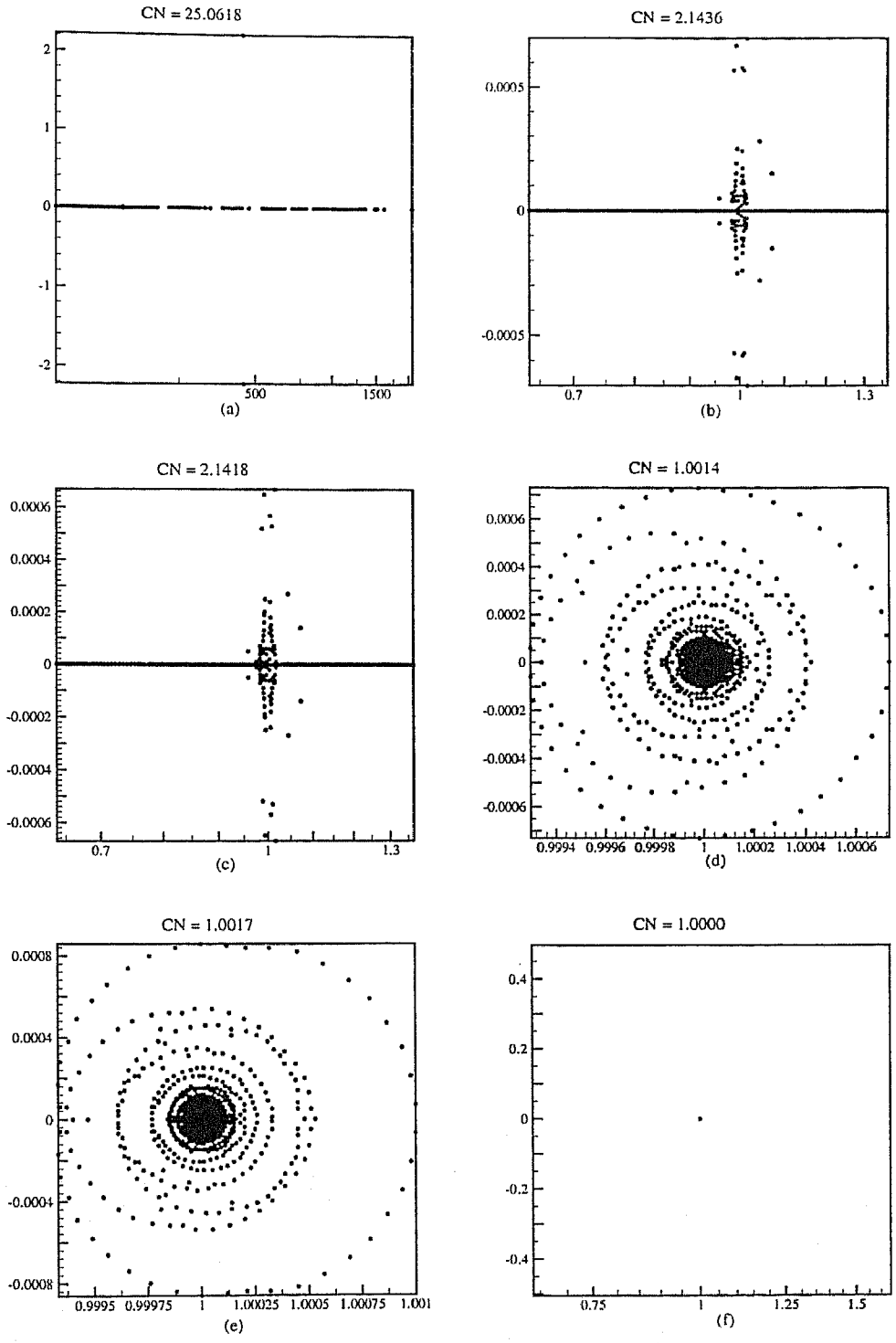


Figure 3.34: Eigenvalue spectrum and condition numbers for Pe = 0.10, using a 125 × 11 grid in the unsteady-nonlinear case.

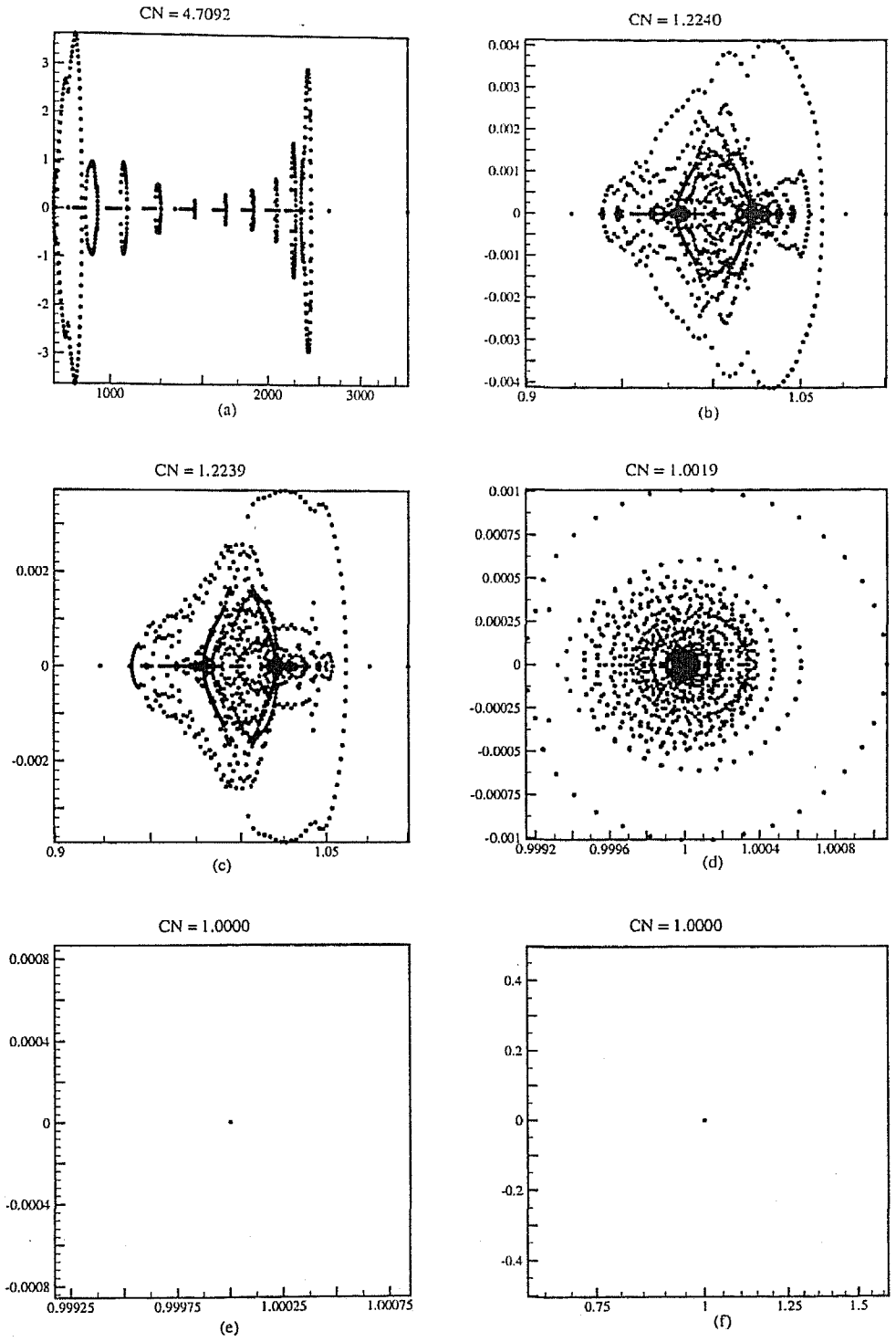


Figure 3.35: Eigenvalue spectrum and condition numbers for Pe = 1, using a 125 × 11 grid in the unsteady-nonlinear case.

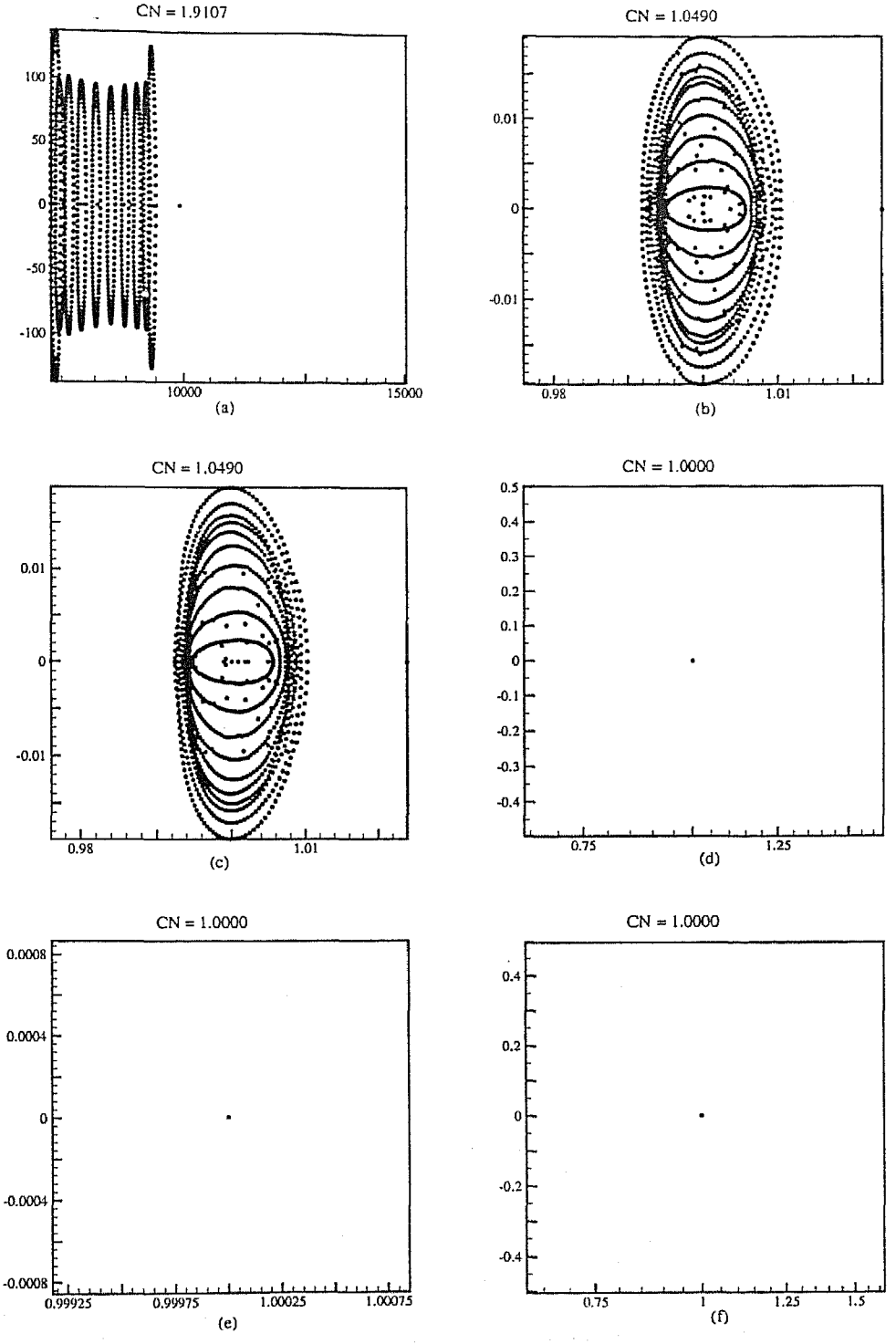


Figure 3.36: Eigenvalue spectrum and condition numbers for Pe = 10, using a 125×11 grid in the unsteady-nonlinear case.

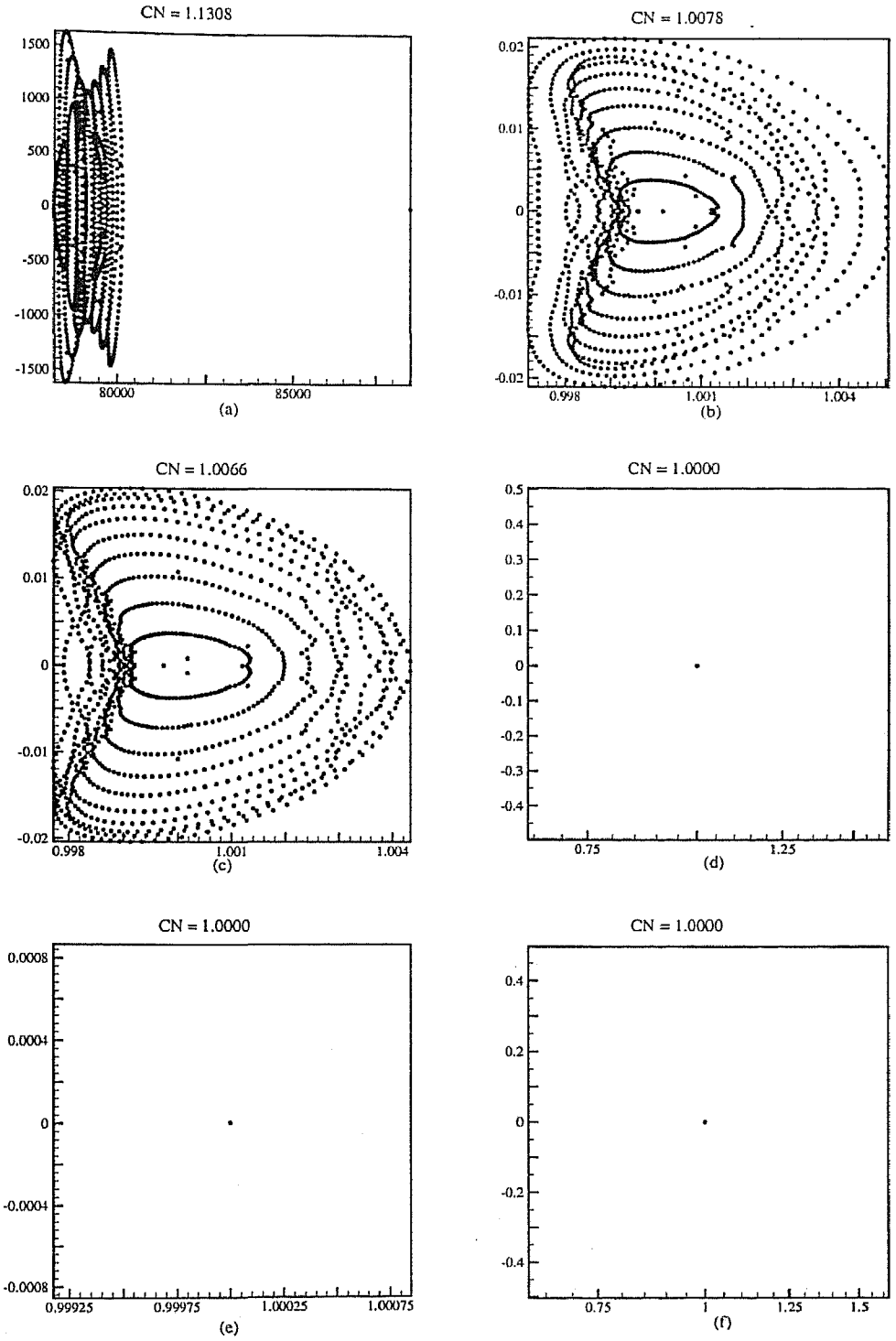


Figure 3.37: Eigenvalue spectrum and condition numbers for Pe = 100, using a 125×11 grid in the unsteady-nonlinear case.

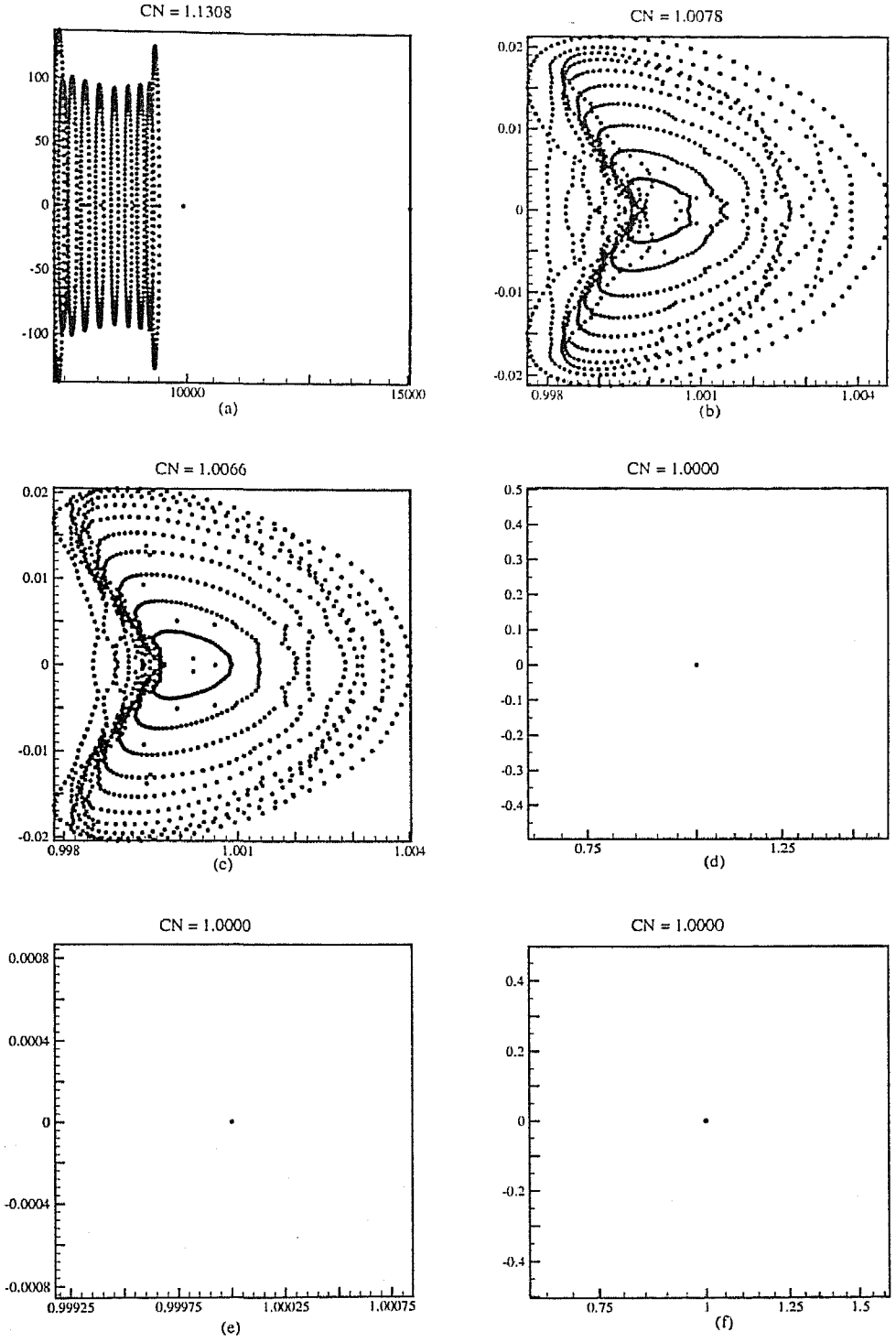


Figure 3.38: Eigenvalue spectrum and condition numbers for Pe = 1000, using a 125×11 grid in the unsteady-nonlinear case.

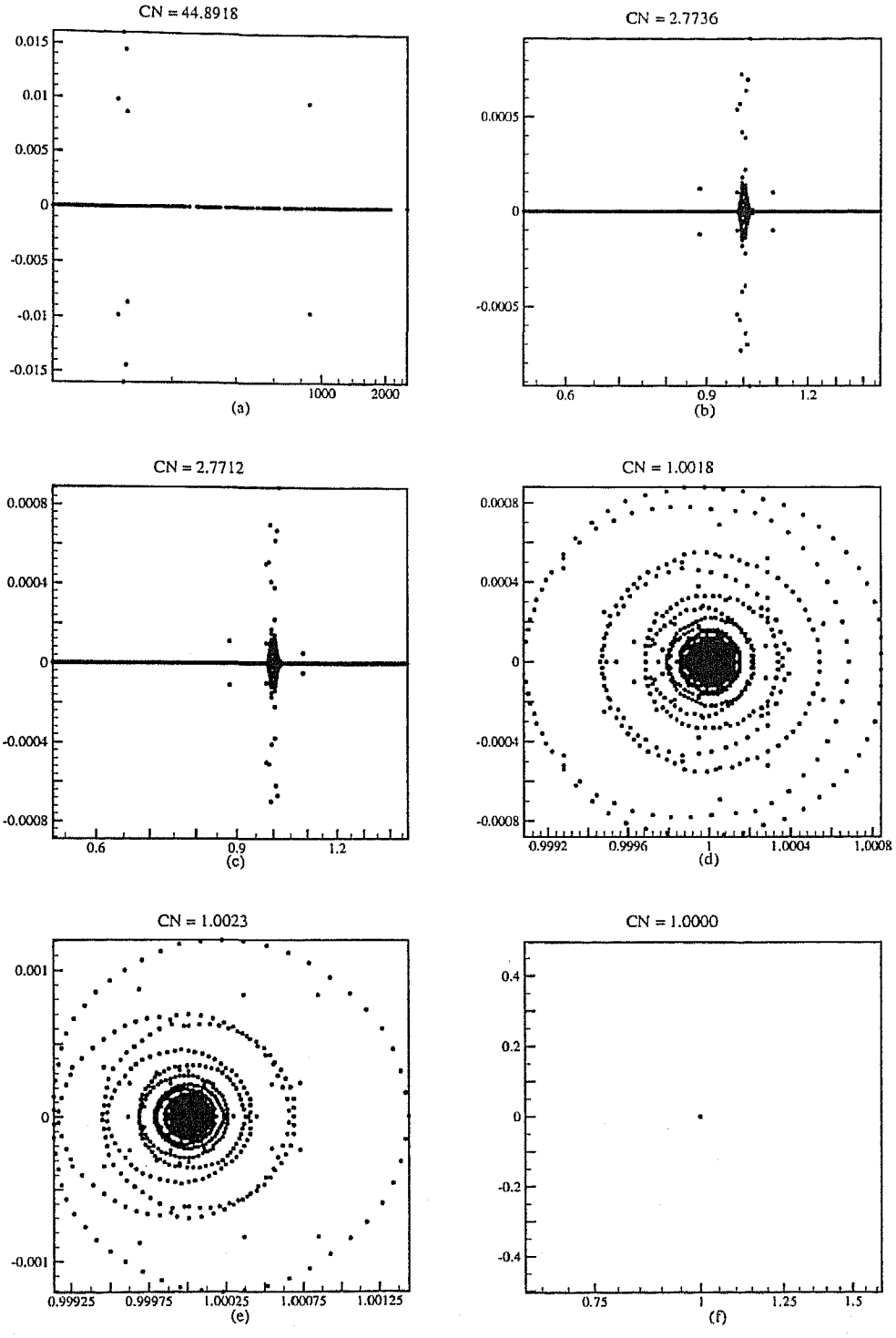


Figure 3.39: Eigenvalue spectrum and condition numbers for Pe = 0.10, using a 151 × 15 grid in the unsteady-nonlinear case.

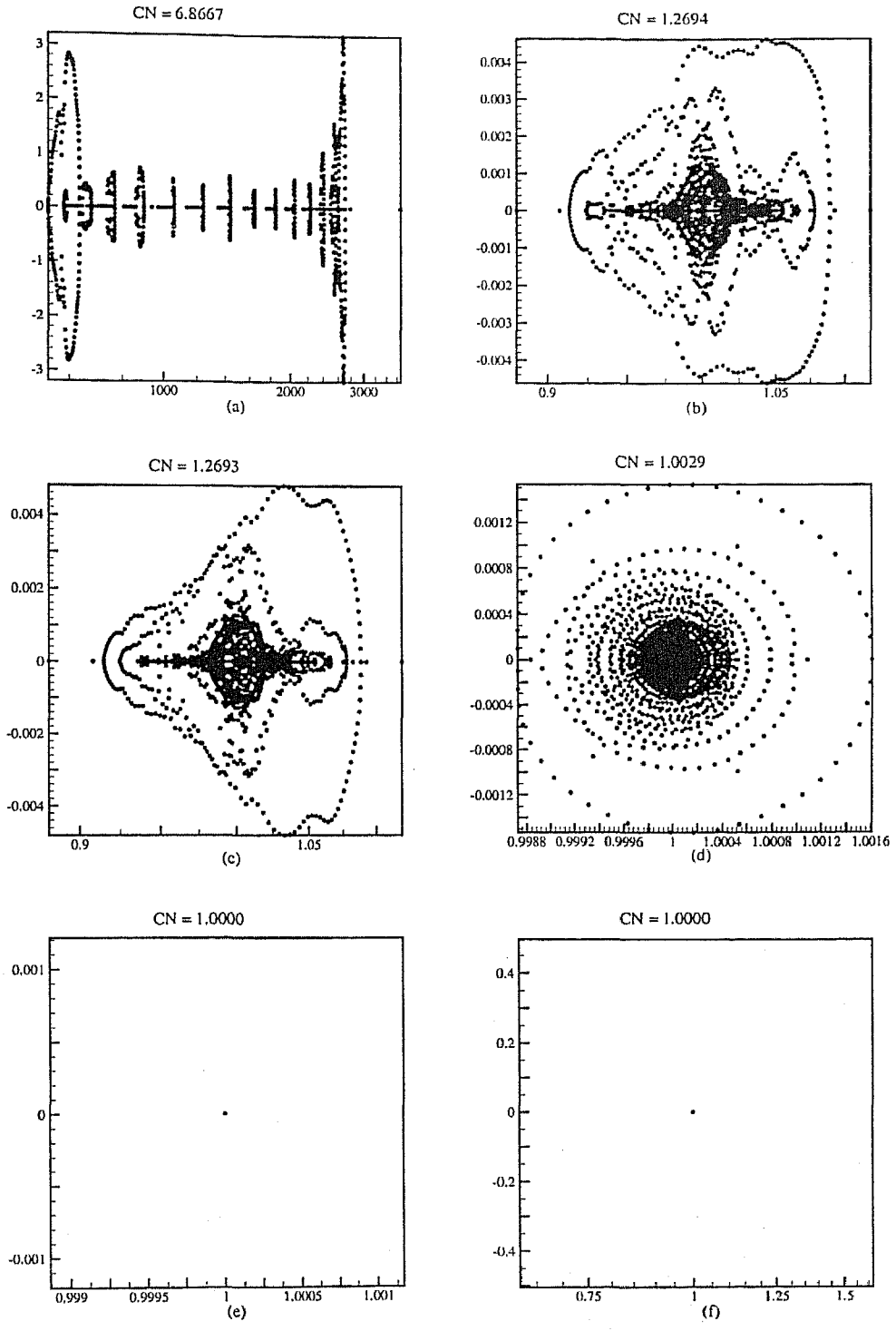


Figure 3.40: Eigenvalue spectrum and condition numbers for Pe = 1, using a 151×15 grid in the unsteady-nonlinear case.

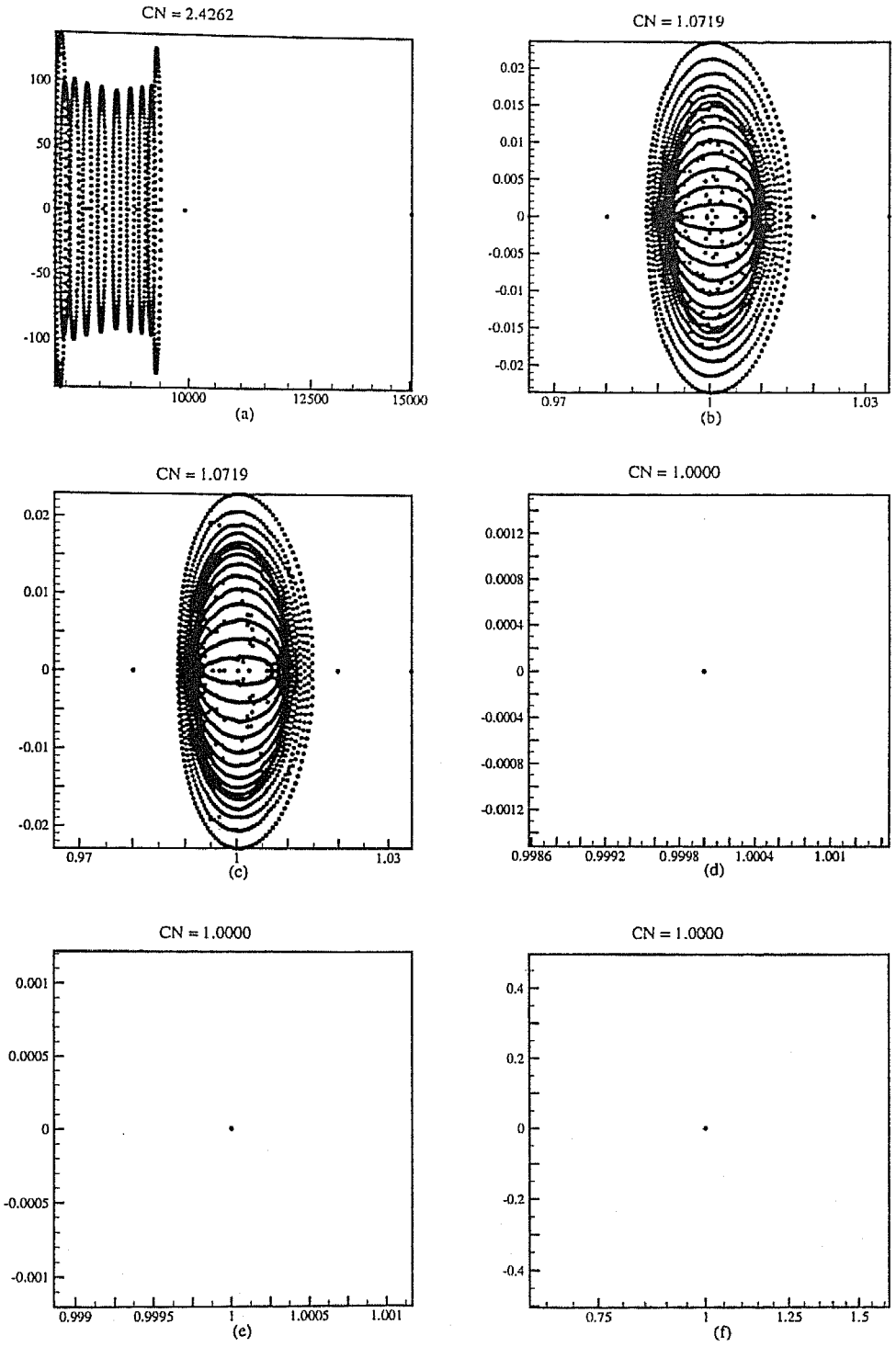


Figure 3.41: Eigenvalue spectrum and condition numbers for Pe = 10, using a 151 × 15 grid in the unsteady-nonlinear case.

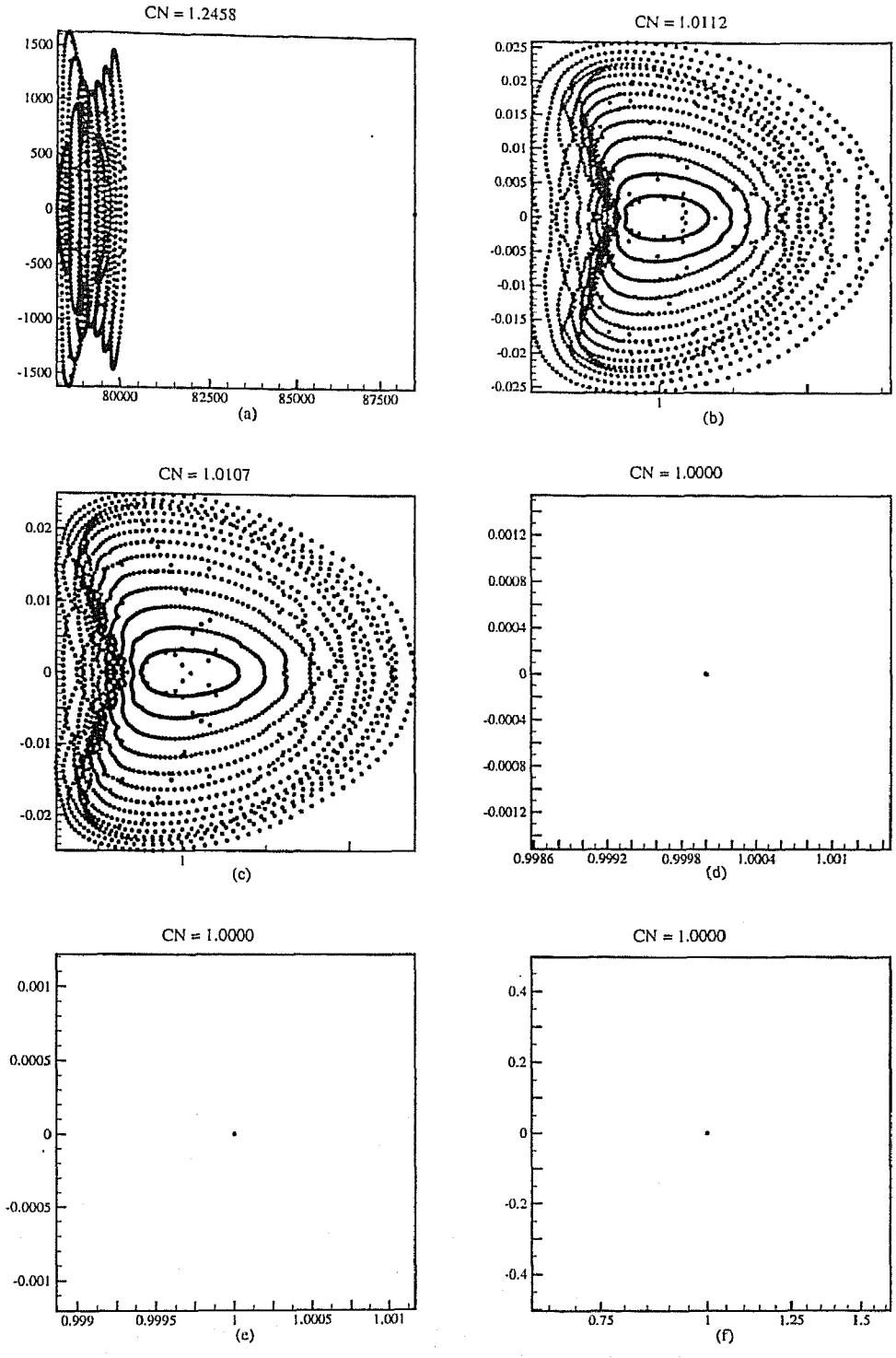


Figure 3.42: Eigenvalue spectrum and condition numbers for Pe = 100, using a 151 × 15 grid in the unsteady-nonlinear case.

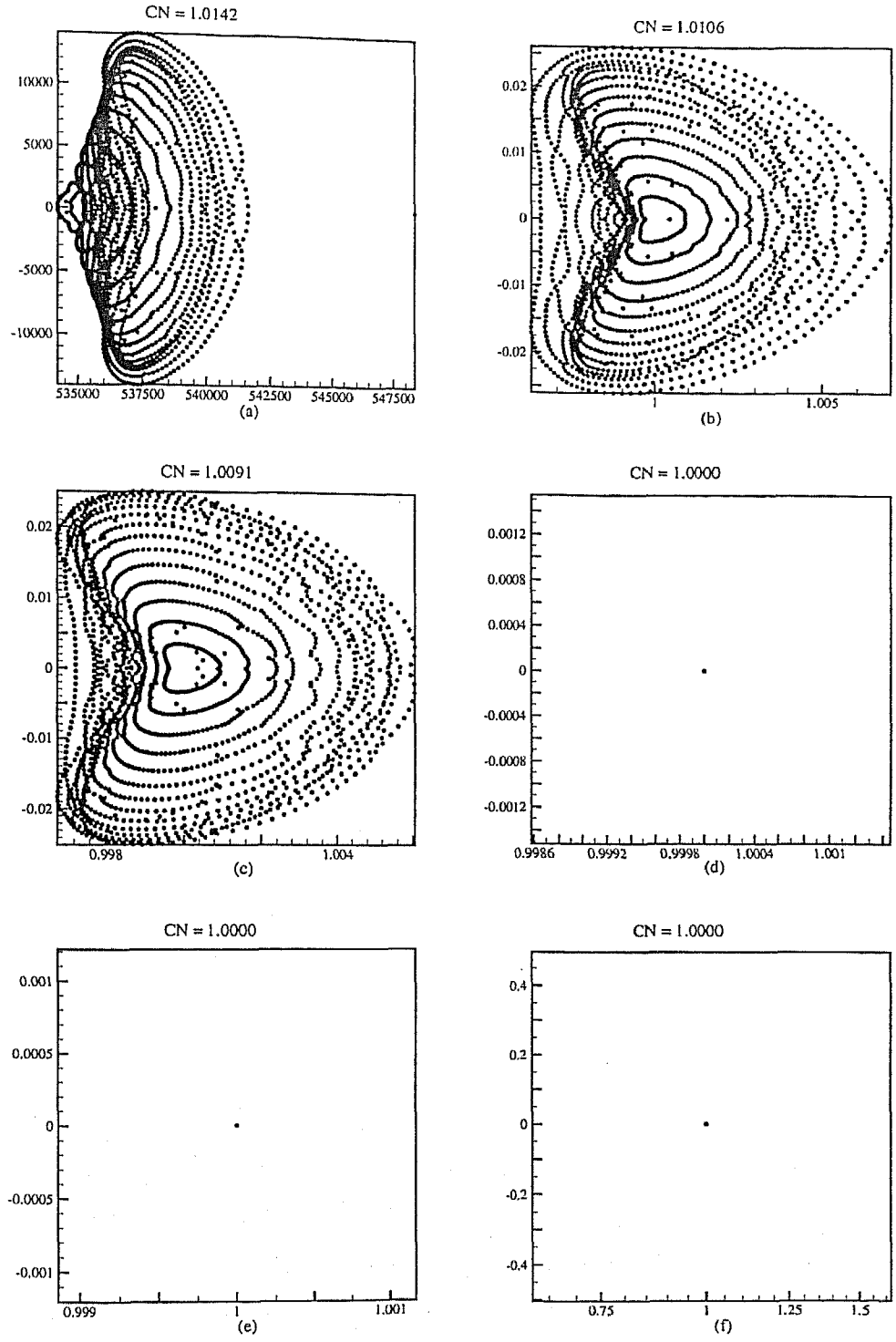


Figure 3.43: Eigenvalue spectrum and condition numbers for Pe = 1000, using a 151×15 grid in the unsteady-nonlinear case.

From the results presented in this study, the following observations can be recorded:

1. As shown in Figure 3.10, condition-number for the original matrix in the steady-linear problem, initially decreases with an increase in Peclet number (for Peclet number 0.10 to 100). Later, it starts increasing with an increase in Peclet number (for Peclet number > 500). In between, for Peclet number 100 to 500, a transition from decreasing to an increasing condition-number trend takes place. Condition-number for the original matrix in the unsteady-nonlinear problem (Figure 3.11), decreases almost asymptotically with Peclet number and then, finally becomes constant with a magnitude one.

   Here, the mathematical nature of the partial differential equation being solved depends on Peclet number. As the Peclet number increases the problem changes from elliptic to hyperbolic. Since we are solving it as an elliptic problem for all values of Peclet number, obtaining the solution becomes difficult for higher values of Peclet numbers where the problem is actually hyperbolic. Rate of convergence in our solution procedure validates this point.

   Condition numbers represent the mathematical complexity in solving the problem. In Figure 3.10 and Figure 3.11, when the condition is decreasing with increasing Peclet number, mathematically the problem is changing from an elliptic to relatively simple hyperbolic problem. But of course, as observed from the convergence experience, practically solving the problem is becoming increasingly difficult with increasing Peclet number.

2. Eigenvalue spectra shown in Figure 3.12 and Figure 3.12 clearly show an increase in the scatter of eigenvalues with increase in Peclet number. Therefore, the relatively poor convergence of iterative methods for higher values of Peclet numbers is explained.

3. In both the problems considered, condition number increases with a decrease in grid spacing for a given value of Peclet number (Figure 3.10 and Figure 3.11). This effect is relatively less prominent at greater values of Peclet number especially for the unsteady-nonlinear problem. It has been

further observed that, although the eigenvalue spectrum for these matrices remains approximately same on finer grids, a few eigenvalues move closer and closer to zero. This effect on finer grids results in an increased condition-number and gives rise to poor convergence of iterative solvers.

4. Effect of ILU-preconditioning is well seen for both the problems considered, Figures 3.14 to 3.28 for the steady-linear problem and from Figures 3.29 to 3.43 for the unsteady-nonlinear problem. As the width of the band considered during LU decomposition is increased, the eigenvalue spectrum improves (*i.e.*, the spread of eigenvalues decreases) and condition number also decreases. This observation can be explained from the fact that as the number no diagonals considered during LU decomposition increases, error introduced during LU decomposition decreases and the decomposition becomes more and more exact. Therefore, the preconditioned matrix $L^{-1}AU^{-1}$ becomes more closer to the identity matrix I.

Chapter 4

Mathematical Formulation for Modeling Transport Phenomena in Czochralski Crystal Growth Processes

Czochralski crystal growth is a physically complex and mathematically rich process. The quality of grown crystals depends on many macroscopic and microscopic physical effects (Figure 4.1). Formulating a mathematical model to incorporate all these effects is not an easy task and judicious choices need to be made in this regard. Mathematical model presented here allows the consideration of as many effects as the problem demands and is believed crucial for the modeling process. This model draws heavily from the earlier works of Zhang *et al.* [1995], Prasad *et al.* [1997] and Zou *et al.* [1997] and ongoing Ph.D. thesis of Banerjee [2004].

Figure 4.2 shows a schematic of the right-half of the considered axisymmetric Cz-system. The domain consist of various materials in different phases with significantly different thermophysical and transport properties (Table 4.4). The Cz-model presented here focuses on the growth of Nd:YAG (Neodymium doped Yttrium Aluminum Garnet) single crystals. Nd:YAG falls in the category of oxide crystals, and since, the melt phase for the growth of these crystals does not contain any volatile component, need for an encapsulant layer over the melt gets eliminated.

An introduction to the basic growth process and various transport mechanisms that take place inside a Cz-system is given in Section 1.2 of Chapter 1. A list of various important process parameters is given in Table 4.3.

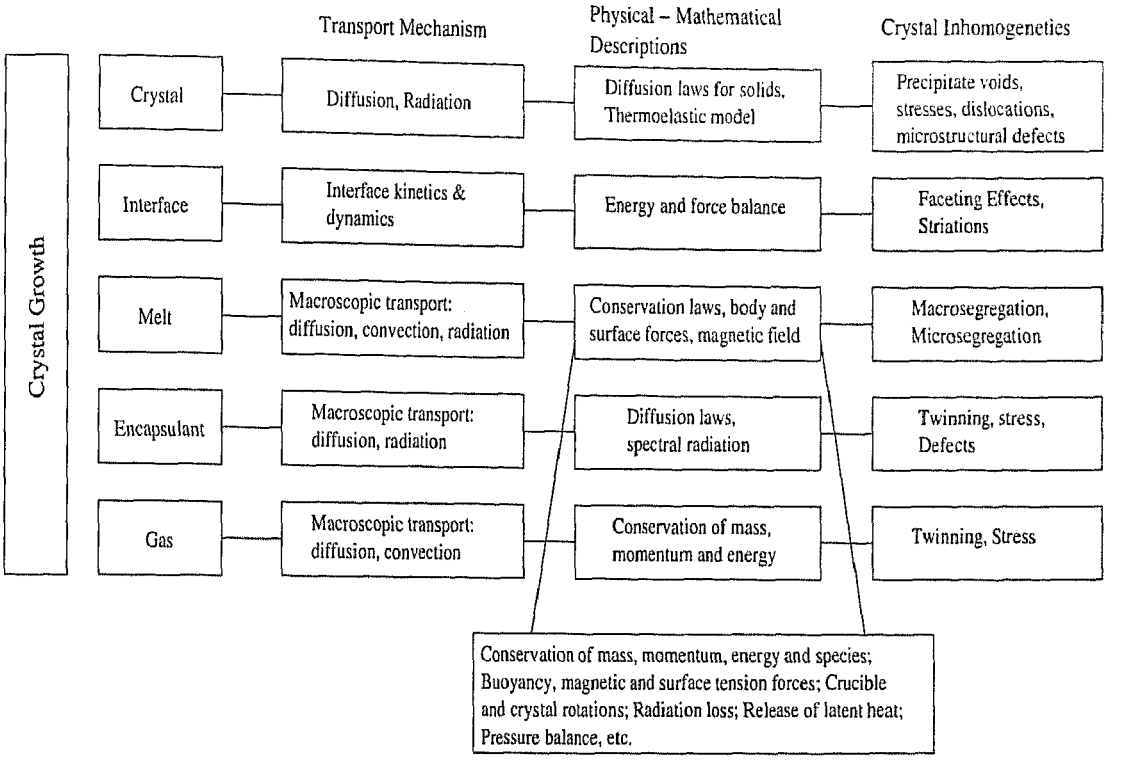


Figure 4.1: Transport mechanism and inhomogeneities associated with crystal growth processes.

4.1 Equations governing flow, heat transfer and solute transport

The assumptions that go into developing the mathematical model for flow, heat transfer and solute transport inside the considered axisymmetric system are as follows:

1. The fluid flow is two-dimensional, laminar and incompressible.
2. The fluids are Newtonian.
3. Thermophysical properties are constant and uniform in various phases.
4. Boussinesq approximation for buoyancy-driven convection is applicable.
5. Viscous dissipation is negligible.
6. Free surface deformation is negligible, *i.e.*, free surface is assumed to be flat.

7. Change in surface tension of the liquid phase with temperature is negligible, *i.e.*, Maragoni convection is absent.

8. Liquid phase is treated as a dilute solution of the solute in the melt.

9. Segregation coefficient is independent of the local growth rate of the crystal and other operating conditions.

10. No diffusion takes place in the solid phase.

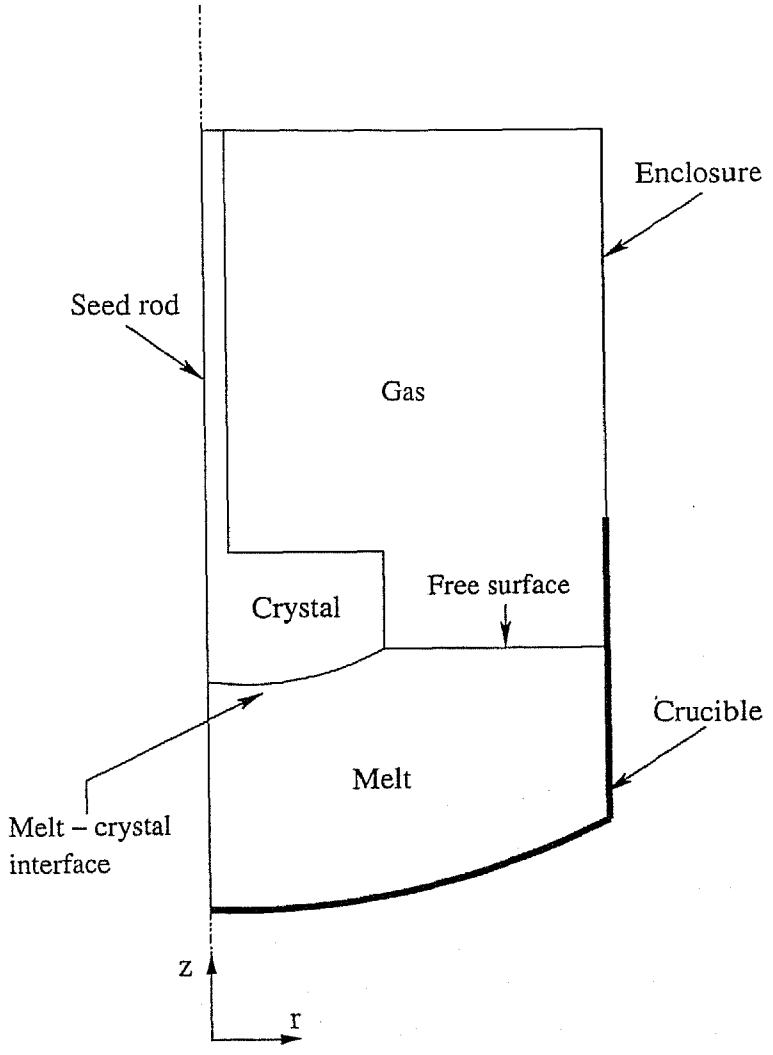


Figure 4.2: Schematic of the right-half of the axisymmetric Cz-system considered, showing various zones, interfaces and the free surface.

The scales that are used to nondimensionalize the governing equations are given in Table 4.1. Where $\bar{\psi}$ refers to the dimensionless thermophysical proper-

Table 4.1: Scales for Nondimensionalization.

Variable	Nondimensionalization
length	$(x, y) = (x^*, y^*)/L$
time	$t = t^*/(L/U_o)$
velocity	$(u, v) = (u^*, v^*)/U_o$
temperature	$\theta = (T - T_f)/(T_h - T_f)$
pressure	$P = (p^* - \rho g x^* - p_\alpha)/(\rho {U_o}^2)$
concentration	$C = C^*/C^*_{ref}$
thermophysical properties	$\bar{\psi}_i = {\psi_i}^*/\psi^*_l$

4.2 Generalized conservative form of the transport equations

The governing equations presented in Section 4.1 can be further put in the following general mathematical form, Zhang *et al.* [1995].

$$\frac{\partial(r^\gamma \bar{\rho}\phi)}{\partial t} + \frac{\partial}{\partial z}(r^\gamma \bar{\rho} u \phi) + \frac{\partial}{\partial r}(r^\gamma \bar{\rho} v \phi) = \frac{\partial}{\partial z}(r^\gamma \Gamma_\phi \frac{\partial \phi}{\partial z}) + \frac{\partial}{\partial r}(r^\gamma \Gamma_\phi \frac{\partial \phi}{\partial r}) + r^\gamma \mathrm{S}_\phi \quad (4.6)$$

where ϕ is the generalized variable, S is the volumetric source, and Γ is the diffusion coefficient. The index γ is set to zero if Cartesian coordinates are used, and is unity for polar coordinates. Equation 4.6 can be used for the entire multiphase, multicomponent domain with the provision to account for local properties and abrupt changes in transport properties across the zone boundaries, and their possible movements. The values for ϕ, Γ and S for continuity, momentum, energy and concentration equations for an axisymmetric geometry are given in Table 4.2.

Table 4.2: Variables of Equation 4.6 in an axisymmetric geometry.

Equations	Variable ϕ	Diffusivity Γ	Source term S
continuity	1	0	0
z-momentum	u	$\bar{\mu}_i$	$-\partial P/\partial z + \mathrm{Gr}\bar{\beta}_i\bar{\rho}_i\theta$
r-momentum	v	$\bar{\mu}_i$	$-\partial P/\partial r - \bar{\mu}_i v/r^2 + \bar{\rho}_i w^2/r$
rw-momentum	rw	$\bar{\mu}_i$	$-(2\bar{\mu}_i/r)\partial(rw)/\partial r$
θ-energy	$\bar{C}_{pi}\theta$	$\bar{k}_i/(\bar{C}_{pi}\mathrm{Pr})$	0
concentration	C	1/Sc	0

4.3 Initial and boundary conditions

The governing equations given in above sections require both initial and boundary conditions for a complete solution. Detail of these conditions is summarized below.

4.3.1 Initial Conditions

Initial conditions for the set of governing equations depends on the solution strategy adopted to get the complete solution. In case, a steady-state solution is sought any initial field can be taken as the initial condition. However, for transient analysis exact initial conditions need to be used. These aspects are discussed in Chapter 5 along with the solution strategy.

4.3.2 Boundary conditions

Boundary conditions at various boundaries and interfaces in a Cz-system are as follows.

Bottom and side wall of the crucible

Here for the velocities, no-slip boundary condition is applicable. Which can be written as

$$u = v = 0, \quad w = \mathrm{Re}_c r \tag{4.7}$$

and for the energy equation we have specified temperature T_w at these walls and that gives the following nondimensional boundary condition.

$$\theta = 1 \tag{4.8}$$

For the concentration equation free gradient boundary condition is applicable.

$$(\nabla C) \cdot \mathbf{n} = 0 \tag{4.9}$$

Top and side wall of the enclosure

Again for the velocities, no-slip boundary condition is applicable. That is

$$u = v = w = 0 \tag{4.10}$$

and for the energy equation we have specified temperature variation at these walls.

$$\theta = \text{specified variation} \tag{4.11}$$

For the concentration equation free gradient boundary condition is applicable.

$$(\nabla C) \cdot \mathbf{n} = 0 \tag{4.12}$$

Axis of symmetry

For the portion of the axis that is inside melt following conditions are applicable.

$$\frac{\partial u}{\partial r} = 0, \quad v = 0, \quad w = 0 \tag{4.13}$$

$$\frac{\partial \theta}{\partial r} = 0 \tag{4.14}$$

$$\frac{\partial C}{\partial r} = 0 \tag{4.15}$$

Solid portion of the axis of symmetry need only the following.

$$\frac{\partial \theta}{\partial r} = 0 \tag{4.16}$$

Melt-crystal interface

Here for the velocities, no-slip boundary condition is applicable. Which can be written as

$$u = v = 0, \quad w = \mathrm{Re}_s r \tag{4.17}$$

and for the energy equation we have the fusion temperature T_f at these walls and that gives the following nondimensional boundary condition.

$$\theta = 0 \tag{4.18}$$

At the melt-crystal interface, solute conservation gives the following boundary condition.

$$-\frac{1}{\mathrm{Sc}}(\nabla C) \cdot \mathbf{n} = C(1 - k_0)\mathbf{V}_{\mathbf{pull}} \cdot \mathbf{n} \tag{4.19}$$

where k_0 is the segregation coefficient and $\mathbf{V}_{\mathbf{pull}}$ is the nondimensional pull rate. Above boundary condition has been derived by equating the segregation flux at the interface with the diffusion flux just below it in the melt. Detailed discussion of the same can be found in Section 5.8 of Chapter 5.

Free surface

Here with the flat interface assumption, the kinematic and zero shear stress boundary conditions at the free surface gives the following.

$$u = 0, \quad \frac{\partial v}{\partial z} = 0 \tag{4.20}$$

for the swirl velocity

$$\frac{\partial w}{\partial z} = 0 \tag{4.21}$$

A flux balance at the free-surface gives

$$-k\frac{\partial T}{\partial z}\bigg|_l = -k\frac{\partial T}{\partial z}\bigg|_g + \epsilon\sigma_r(T^4 - T_\infty^4) \tag{4.22}$$

which on nondimensionalization gives the following

$$-k\frac{\partial \theta}{\partial z}\bigg|_l = -k\frac{\partial \theta}{\partial z}\bigg|_g + \mathrm{Bi}_{r,lg}(\theta_{lg} - \theta_\infty) \tag{4.23}$$

and for the concentration equation again the following is applicable

$$\frac{\partial C}{\partial r} = 0 \tag{4.24}$$

Top and side wall of the crystal

Here we no slip for the all the velocity components

$$u = U_s, \quad v = 0, \quad w = \mathrm{Re}_s r \tag{4.25}$$

and for temperature, the following flux balance

$$-k\frac{\partial \theta}{\partial z}\bigg|_s = -k\frac{\partial \theta}{\partial z}\bigg|_g + \mathrm{Bi}_{r,s}(\theta_s - \theta_\infty) \tag{4.26}$$

4.4 Process parameters and material properties

Transport phenomena inside the Cz-system, and therefore the crystal quality depends on the effect of a number of system parameters. A list of the various important parameters inside the Cz-system is given below in Table 4.2. Typical values of material properties for Nd:YAG single crystals is given in Table 4.3.

Table 4.3: Various Process parameters in a Cz-system.

Pull rate	Crystal diameter
Initial melt height	Initial crystal height
Enclosure diameter	Enclosure height
Crucible temperature	Enclosure temperature variation
Ambient temperature	Gas pressure
Crystal rotation	Crucible rotation

Table 4.4: Thermophysical properties of Nd:YAG single crystals where subscripts s, l, g and c denote solid-crystal, melt, gas and crucible respectively.

Description	Symbol	Value
Density (solid)	ρ_s	4300 Kg/m^3
Density (liquid)	ρ_l	3600 Kg/m^3
Density (gas)	ρ_g	0.1602 Kg/m^3
Thermal conductivity (solid)	k_s	8 W/(mK)
Thermal conductivity (liquid)	k_l	4 W/(mK)
Thermal conductivity (gas)	k_g	0.139 W/(mK)
Melting point	T_f	2243 K
Melt expansivity	β	1.8×10^{-5} K^{-1}
Viscosity (liquid)	μ_l	4.68×10^{-2} Kg/(ms)
Viscosity (gas)	μ_g	6.93×10^{-5} Kg/(ms)
Heat Capacity (solid)	c_s	800 J/(KgK)
Heat Capacity (liquid)	c_l	800 J/(KgK)
Heat Capacity (gas)	c_g	1419 J/(KgK)
Heat of fusion	ΔH_f	4.55×10^5 J/Kg
Emissivity (solid)	ϵ_s	0.3
Emissivity (liquid)	ϵ_l	0.3
Emissivity (gas)	ϵ_c	0.5

Chapter 5

Numerical solution of the partial differential equations governing Czochralski crystal growth

In this chapter the numerical methodology for solving the generalized conservative form of the governing equations, Equation 4.6, is presented (Banerjee [2004], Zhang *et al.* [1995]). Then, the overall solution approach is discussed along with the details of the calculations for crystal pull rate and solute segregation.

5.1 Coordinate transformation

A physical problem composed of a domain of irregular shape is difficult to solve in a regular orthogonal coordinate system such as Cartesian owing to the fact that the boundaries of such domains may not conform to the regular coordinate axes. Hence an accurate representation of the system geometry becomes difficult. It is in this context that the concept of boundary fitted (body-fitted) coordinate system has evolved. When one use such a coordinate transformation, an irregular geometry in a physical space (such as Cartesian) is transformed to a regular geometry in a computational space wherein boundaries of the transformed geometry conform to the coordinate axes and then calculations are performed in the computational space. This transformation, which depends on the shape of the physical domain of interest, does not necessarily guarantee orthogonality of the grid lines in the transformed plane. In fact in most cases, the transformation results in a generalized nonorthogonal curvilinear system in which the problem is to be solved.

Introduction of nonorthogonality is generally detrimental, as the transformed governing equations contain extra terms to account for non-orthogonality of the coordinate system. However, the fact that an accurate representation of the physical domain is possible by this technique, largely outweighs the above mentioned limitations.

The conservation equation 4.6 in a generalized coordinate system (ξ, η) can be written as:

$$r^\gamma J\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial}{\partial\xi}[\alpha_\xi J_\xi - \beta_\xi J_\eta] + \frac{\partial}{\partial\eta}[\alpha_\eta J_\eta - \beta_\eta J_\xi] = r^\gamma J S_\phi \tag{5.1}$$

where

$$J_\xi = \rho u_\xi \phi - \frac{\Gamma}{h_\xi}\left(\frac{\partial\phi}{\partial\xi}\right)$$

and

$$J_\eta = \rho u_\eta \phi - \frac{\Gamma}{h_\eta}\left(\frac{\partial\phi}{\partial\eta}\right).$$

Other coefficients are

$$\alpha_\xi = r^n h_\xi {h_\eta}^2/J, \qquad \alpha_\eta = r^n h_\eta {h_\xi}^2/J,$$

$$\beta_\xi = r^n \lambda h_\eta/J, \qquad \beta_\eta = r^n \lambda h_\xi/J,$$

$$h_\xi = ({x_\xi}^2 + {y_\xi}^2)^{1/2}, \quad h_\eta = ({x_\eta}^2 + {y_\eta}^2)^{1/2},$$

$$\lambda = x_\xi x_\eta + y_\xi y_\eta, \qquad J = x_\xi y_\eta - y_\xi x_\eta.$$

It is important to note some of the properties of coefficient of α and β. Both these coefficients have units of area, and while α's are always positive; β's may be positive, negative or zero. The absolute magnitude of β_ξ is always less than α_ξ; and if the coordinates (ξ, η) are orthogonal; then $\beta_\xi = 0$, and $\alpha_\xi = h_\eta$. These coefficients can be conceptually regarded as areas. In view of their properties, α's are referred to as the primary areas and β's as secondary areas. The subscripts ξ and η in these coefficients denote their correspondence to a ξ = constant and η = constant lines respectively. With this interpretation, the flux across a finite volume face may also be thought of as composed of two components.

5.2 Grid generation

As is evident, Cz growth systems are composed of zones with vastly different thermophysical properties and internally moving interfaces and free surfaces. To perform accurate simulations for these purposes, the grid system in different zones need to be generated in such a way that the interfaces separating these zones are preserved and coincide with some grid lines, permitting grid points to move only along these interfaces. Here for grid generation, MAGG (Multizone Adaptive Grid Generation) scheme developed by Zhang *et al.* [1995] has been used. The scheme has been developed based on the variational method and minimizes an integral function which is a measure of grid characteristics, namely the smoothness, orthogonality, weighted cell area and inertia of the grids. The scheme allows grids to move adpatively as the solutions progress, domains change, or both. Moreover, the grids are concentrated in the regions of large variations in field variables using appropriate weighting functions.

5.3 Finite volume discretization

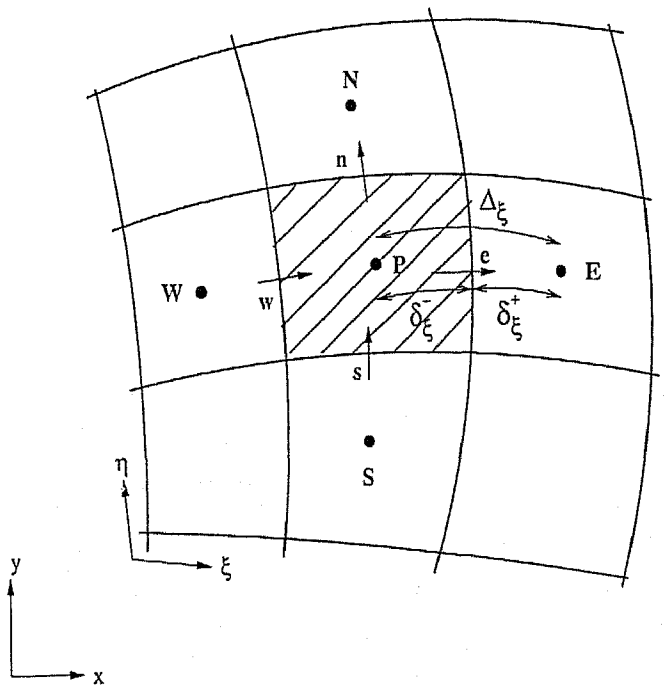


Figure 5.1: Curvilinear finite-Volume grid arrangement.

The grids used are a structured trapezoidal mesh, as shown in Figure 5.1. For a typical point P, the generalized conservation equation, Equation 5.1, can be discretized in the following manner.

$$\begin{aligned}\frac{(r^\gamma J\rho\phi)_P - (r^n J\rho\phi)^o{}_P}{\Delta t}\Delta\xi\Delta\eta + (\alpha_\xi J_\xi - \beta_\xi J_\eta)_e\Delta\eta - (\alpha_\xi J_\xi - \beta_\xi J_\eta)_w\Delta\eta \\ +(\alpha_\eta J_\eta - \beta_\xi J_\xi)_n\Delta\xi - (\alpha_\eta J_\eta - \beta_\eta J_\xi)_s\Delta\xi = (r^\gamma J\hat{S})_P\Delta\xi\Delta\eta\end{aligned} \tag{5.2}$$

Above equation represents an overall conservation of ϕ in a finite volume in terms of its fluxes at the auxiliary nodes, and is referred to as the flux discretization equation. The source term $\hat{S}$ is the original source term plus the terms consisting of the grid velocity components,

$$r^\gamma J\hat{S} = r^\gamma JS + \frac{\partial\rho r^\gamma u_g\phi}{\partial\xi} + \frac{\partial\rho r^\gamma v_g\phi}{\partial\eta} \tag{5.3}$$

where the grid velocities are defined as

$$u_g = \frac{\partial y}{\partial\eta}\frac{\partial x}{\partial t} - \frac{\partial x}{\partial\eta}\frac{\partial y}{\partial t} \qquad\qquad v_g = \frac{\partial x}{\partial\xi}\frac{\partial y}{\partial t} - \frac{\partial y}{\partial\xi}\frac{\partial x}{\partial t} \tag{5.4}$$

Although in the calculation grids velocity components have been neglected and $\hat{S} = S$ is used.

Now to obtain the finite difference form of Equation 5.2 in terms of ϕ at the centers of the finite volumes, Power Law scheme of Patankar [1980] is used. The fluxes using this scheme can be represented as follows.

$$\begin{aligned}(\alpha_\xi J_\xi)_e &= F_e\phi_P + [D_e A(|Pe_e|) + [|\ -F_e, 0\ |]](\phi_P - \phi_E) \\ (\alpha_\xi J_\xi)_w &= F_w\phi_P + [D_w A(|Pe_w|) + [|\ 0, F_w\ |]](\phi_W - \phi_P) \\ (\alpha_\eta J_\eta)_n &= F_n\phi_P + [D_n A(|Pe_n|) + [|\ -F_n, 0\ |]](\phi_P - \phi_N) \\ (\alpha_\eta J_\eta)_s &= F_s\phi_P + [D_s A(|Pe_s|) + [|\ 0, F_s\ |]](\phi_S - \phi_P)\end{aligned} \tag{5.5}$$

where

$$D_e = \frac{\Gamma_e}{h_{\xi_e}}, \quad F_e = (\rho u_\xi)_e, \quad Pe_e = \frac{F_e}{D_e}, \quad \Gamma_e = \frac{\Delta_\xi}{(\delta_\xi^-/\Gamma_P + \delta_\xi^+/\Gamma_E)} \tag{5.6}$$

and the function $A(|Pe_e|)$ is defined by Patankar [1080] as

$$A(|Pe_e|) = \max\left[0,\ \ 1 - 0.1|Pe|^5\right] \tag{5.7}$$

The other coefficients along faces w, n, s are similar to Equation 5.6. Substituting these fluxes from Equation 5.5 into Equation 5.2 we obtain the following discretized governing equation in the ξ, η coordinates.

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b \tag{5.8}$$

where

$$a_E = [D_eA(|Pe_e|) + [|\ -F_e, 0\ |]]\,(\phi_P - \phi_E), \tag{5.9}$$

$$a_W = [D_wA(|Pe_w|) + [|\ 0, F_w\ |]]\,(\phi_W - \phi_P), \tag{5.10}$$

$$a_N = [D_nA(|Pe_n|) + [|\ -F_n, 0\ |]]\,(\phi_P - \phi_N), \tag{5.11}$$

$$a_S = [D_sA(|Pe_s|) + [|\ 0, F_s\ |]]\,(\phi_S - \phi_P), \tag{5.12}$$

$$a_P^o = \frac{\rho^o(Ja)_P^o}{\Delta t}, \tag{5.13}$$

$$a_P = a_E + a_W + a_N + a_S + a_P^o - S_PJ\Delta\xi\Delta\eta \tag{5.14}$$

The remaining terms from the substitution are included in b. In order to identify the origin of these terms b, is expressed as:

$$b = b_s + b_{no} \tag{5.15}$$

where

$$b_s = a_P^o\phi_P^o + S_cJ\Delta\xi\Delta\eta \tag{5.16}$$

and

$$b_{no} = [(\beta_\eta J_\xi)_n - (\beta_\eta J_\xi)_s \Delta\xi] + [(\beta_\xi J_\eta)_e - (\beta_\xi J_\eta)_w \Delta\xi] \\ -(F_e - F_w)\phi_P \Delta\eta - (F_n - F_s)\phi_P \Delta\xi \tag{5.17}$$

In the above discretized formulation the source term is linearized as $S_\phi = S_c + S_P\phi_P$. Terms corresponding to the grid velocity components (namely, the rate of change of grid position with time) have been neglected.

5.4 Treatment of pressure-velocity coupling

Staggered grids have commonly been used for transport problems and are considered to be more appropriate for pressure-velocity coupling. However, there are many advantages of using a non-staggered grid system for the present types of problem. For example, the imposition of interfacial boundary conditions in a non-staggered grid is more appropriate. The only drawback of this approach is that it requires a higher order interpolation to calculate fluxes and handle pressure-velocity coupling. The solution algorithm used for fluid flow calculations in the general curvilinear coordinate system is basically similar to the SIMPLER algorithm Patankar [1980], which consist of solving a pressure pressure equation to obtain the pressure field and solving a pressure-correction equation to correct the predicted velocities. However the scheme is more complicated because the velocity directions change continually along the coordinate lines.

5.5 Overall solution scheme for solving the complete set of equations

The overall solution scheme for solving the complete set of governing equations for Czochralski crystal growth is as follows. The growth process is simulated by a series of quasi-static solutions for flow and temperature fields and the solute transport equation is solved in a time marching way for the whole growth period. This approach is based on the observation that the variation of flow and temperature fields is fairly slow with the growth process and therefore, a quasi-static solution shall catch the basic features of the process. On the other hand, the solute partitioning at the melt-crystal interface and its transport in the melt are

intrinsically dynamic processes which are strongly time dependent. Therefore, a transient solution need to be obtained to track the solute partitioning and its redistribution in the melt during growth. The quasi-static solution is obtained for the flow field at a selected crystal height and then, based on this flow field, the concentration equation is solved in a time marching way. By performing a series of such quasi-static calculations, a period of growth process can be simulated and influences of various related factors can be investigated.

5.6 Numerical treatment of moving interfaces

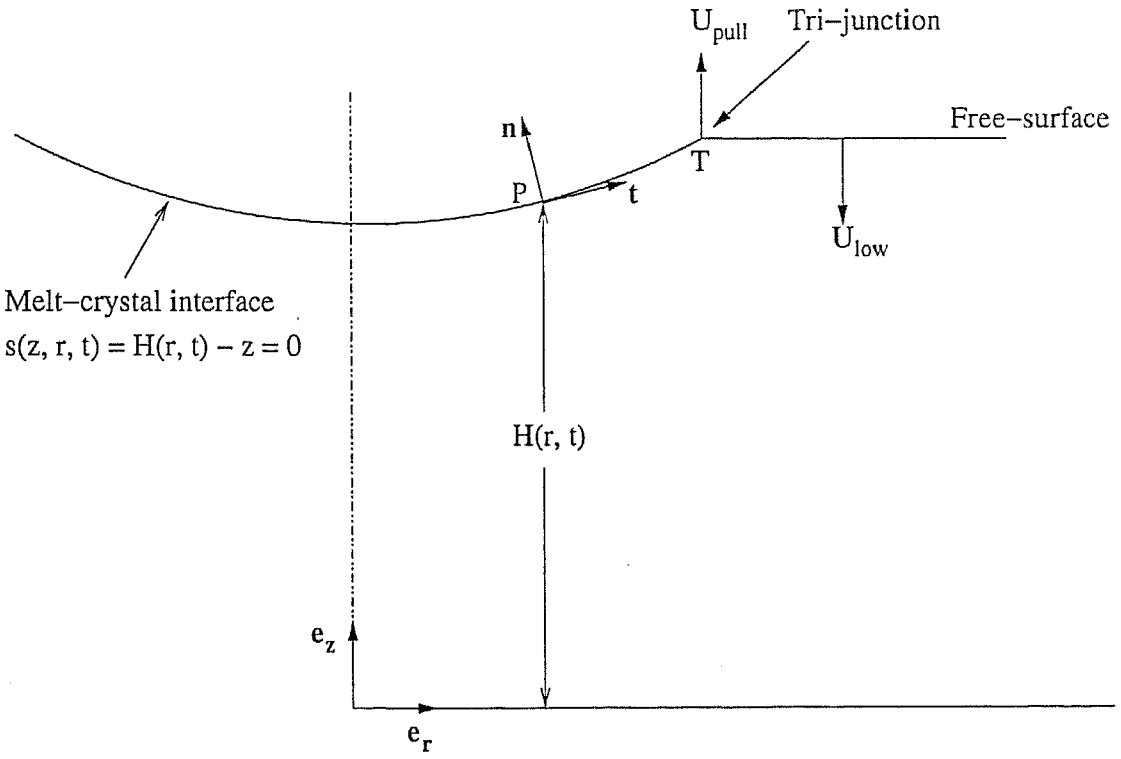


Figure 5.2: Movement of melt-crystal interface and free-surface.

Along with the solution of the field equations (mass, momentum and energy), motion of the melt-crystal interface and the free-surface need to be determined. Solidification of the melt is modeled as a pure substance with a fixed fusion temperature T_f. This implies that the solid and liquid phases are separated by a sharp interface, $s(z,r,t) = H(r,t) - z = 0$, where H is the dimensionless height of the melt-crystal interface, Figure 5.2. Energy balance at the interface (Stefan

condition) defines its position and motion and can be written as

$$\left(\frac{\partial H}{\partial t} - U_{pull}\right)(\mathrm{e_z}\cdot\mathrm{n}) = \frac{\mathrm{Ste}_l}{\mathrm{Pr}_l}\left(\bar{k_s}\frac{\partial\theta_s}{\partial n} - \frac{\partial\theta_l}{\partial n}\right) \tag{5.18}$$

where U_{pull} is dimensionless crystal pull velocity in z-direction.

Pull velocity calculation is based on the assumption that the crystal and melt are not separated at the tri-junction. U_{low} is the dimensionless free-surface velocity in $-\mathrm{e_z}$ direction and U_{sol} is the dimensionless solidification velocity in $-\mathrm{n}$ direction, Figure 5.2. Now, melt-crystal attachment assumption gives the following at the tri-junction.

$$[U_{sol}(\mathrm{e_z}\cdot\mathrm{n})]|_T = (U_{low})|_T + (U_{pull})|_T \tag{5.19}$$

Overall mass conservation for solidification gives the following condition.

$$U_{low}|_T = \mathrm{Rr}^2\bar{\rho}_s\,[U_{sol}(\mathrm{e_z}\cdot\mathrm{n})]|_T \tag{5.20}$$

where Rr is the radius ration (crystal-to-crucible). Solving above three equations gives the following expression for the crystal pull velocity.

$$U_{pull} = -(1-\bar{\rho}_s\mathrm{Rr}^2)\left\{\frac{\mathrm{Ste}_l}{\mathrm{Pr}_l}\left(\bar{k_s}\frac{\partial\theta_s}{\partial n} - \frac{\partial\theta_l}{\partial n}\right)(\mathrm{e_z}\cdot\mathrm{n})\right\}\Bigg|_T \tag{5.21}$$

5.7 Solute segregation

During the crystal growth process, at the melt-crystal interface, there is a tendency for some of the solutes to remain in the melt and for the others to prefer the solid. This phenomenon causes the concentration of the solutes to accumulate or decrease and is termed as *solute segregation*. However, convection in the melt produces mixing and alters the characteristics of the diffusion layer adjacent to the interface. The spatial structure and intensity of the flow determine the

axial and lateral (perpendicular to the growth direction) profiles of the solute concentration in the crystal.

Solute concentration of the solid phase is different from the liquid phase due to segregation. The equilibrium segregation during the solidification of a binary system can be determined from the corresponding phase diagram. In many cases, for low solute concentration the solidus and liquidus curves on a phase diagram can be considered as straight lines in the vicinity of melting temperature. This implies that the ratio of the solubility of a solute in the solid C_s, to that in the melt C_l, remains constant over a concentration range. This ratio is referred as the equilibrium segregation coefficient:

$$k_0 = \frac{C_s}{C_l} \tag{5.22}$$

Segregation coefficient for the considered Nd:YAG system is ≈ 0.2, Lu *et al.* [2000].

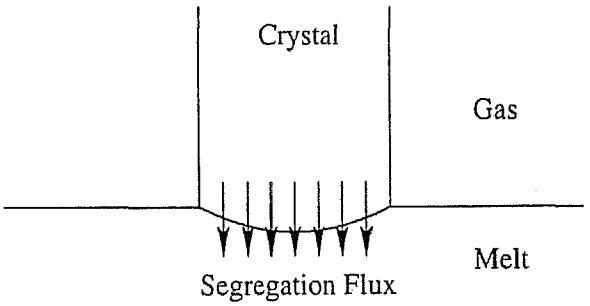


Figure 5.3: Solute Segregation Through the Melt-Crystal Interface.

5.8 Solution of the solute transport equation

During the transient solution of the solute transport equation, initial dimensionless concentration of Nd in the melt is taken as initial condition and boundary conditions used are as given in Section 4.3 of Chapter 4. Further elaboration on the boundary condition at the melt-crystal interface is as follows.

Actually here the argument is that if a melt of solute concentration C^* solidifies then according to the definition of equilibrium segregation coefficient, a solid crystal of concentration $k_0 C^*$ will be formed. Since k_0 is less than one, some

of the solute must come back in the melt through melt-crystal interface. This phenomenon is known as solute segregation and segregation flux at melt-crystal interface can be calculated using Fick's law of diffusion. Thus in dimensional form we have the following equation.

$$-D(\nabla C^*) \cdot \mathrm{n} = C^*(1 - k_0)\mathrm{V}^*_{\mathrm{pull}} \cdot \mathrm{n} \tag{5.23}$$

and this on non-dimensionalization gives the following.

$$-\frac{1}{\mathrm{Sc}}(\nabla C) \cdot \mathrm{n} = C(1 - k_0)\mathrm{V}_{\mathrm{pull}} \cdot \mathrm{n} \tag{5.24}$$

where k_0 is the segregation coefficient and $\mathrm{V}_{\mathrm{pull}}$ is the nondimensional pull rate.

Two major problems are encountered during the solution of the solute transport equation . The first is the implementation of the above segregation boundary condition at the melt-crystal interface. In a solidification process, solute is rejected into the melt if the segregation coefficient is less than unity. This increases the solute concentration in the melt near the interface. To address this issue, the approach of introducing an internal source term to simulate the segregation phenomenon has been followed. The solute is rejected as a flux, $(1 - k)V_{pull}C$ and we have converted this flux to a volumetric source in the first control volume cells in the melt near the interface, such that for any of such cells:

solute input rate due to segregation	=	solute generation rate within the cell due to the source term

This gives source term strength within a cell as

$$S_{cell} = C(1 - k_0)V_{pull} \times \frac{A}{vol} \tag{5.25}$$

where A is the non-dimensional area, vol is non-dimensional volume of the cell and V_{pull} is the vertical component of pull rate, $\mathrm{V}_{\mathrm{pull}}$. The possible error that is introduced because of this approach is minimal in this model since the grid height near the interface is very small due to adaptive nature of the grid and its clustering.

Another issue in solving the mass transport equation is to control the total mass in the system. During the growth, the melt domain continuously drops as more and more melt is solidified into the crystal. This is treated by simply regenerating a new grid system in the changed domain. Therefore it cannot guarantee the total mass conservation after each new grid system is generated. In addition, concentration interpolation is required to start calculations on the new grid. Although a very little imbalance is introduced each time, a large effect in mass deficit may be produced over a long growth period because of the accumulating effect. To avoid this difficulty the approach of conserving solute mass by proportionally correcting the solute concentrations in each grid has been followed.

5.9 Code Validation and Grid Independence

The computer code has been extensively validated, for example against the numerical results of Kobayashi [1978] and Prasad *et al.* [1997]. These code validation and grid independence studies have been reported in the progress report of BRNS project entitled, "Mathematical Modeling of the CVD and Czochralski Crystal Growth Systems, Role of Magnetic Fields, and their Effects on Thermo-mechanical Properties". Grid independent solutions were obtained on a 180×60 mesh.

5.10 Results and Discussion

Using the solution scheme discussed in Sections 5.5 and 5.8 following numerical results have been obtained for the growth of Nd:YAG single crystals.

1. Quasi-static temperature and flow fields in the melt at three melt heights and crystal rotations (Figure 5.7-Figure 5.9).

2. Variation in pull-velocity of a continuously growing crystal for different melt heights and crystal rotations (Figure 5.10).

3. Variation in concentration field in the melt with time for different crystal rotations (Figure 5.11-Figure 5.16).

4. Radial variation in solute concentration at different times for different crystal rotations (Figure 5.17-Figure 5.18).

Moreover, the following values of various important dimensionless parameters have been used.
Crucible radius = 1;
Enclosure radius = 1;
Crystal radius = 0.5;
Initial crystal-height = 0.1;
Total height of the domain in axial direction = 3.0;
Initial melt-height = 1;
Crucible rotation = 0;
Crucible temperature = 1;
Ambient temperature = −5.06.
The enclosure wall temperature drops linearly from crucible temperature to ambient temperature towards top.

The results have been obtained using Preconditioned Conjugate Gradient (PCG) as the linear system solver. Results obtained were found to be exactly identical to the Gauss-Seidel as well as the Line-by-line TDMA iterative solvers.

Figures 5.7 to 5.9 show the effect of crystal rotation on buoyancy driven convection in the melt. When there is no crystal rotation (case (a) in Figures 5.7 to 5.9), flow is completely driven by buoyancy and counter clockwise convective rolls are initiated along the crucible wall. Superposition of crystal rotation with buoyancy driven natural convection (case (b) and (c) in Figures 5.7 to 5.9) brings a two cell pattern in the flow field. As the crystal rotation rate increases, the clockwise convective rolls initiated due to crystal rotation push the natural convective currents towards the wall. Pumping action of crystal rotation makes the hot fluid near bottom move up towards crystal and thus isotherms in the temperature field shift upward towards the crystal.

In Figure 5.10, variation in pull velocity of a continuously growing Nd:YAG crystal for two different initial melt heights is shown. Both the plots show a continuous decrease in pull velocity with time until it becomes zero. As the crystal grows, melt height goes down. In these simulations, this lowering in melt height is very-very small as compared to the increase in crystal height. Thus

during flow and temperature fields in the melt changes almost negligibly while radiation and convection losses from crystal surface continuously increase. At the melt crystal interface, melt side heat flux remains almost same while solid side flux increases continuously with increasing crystal height. This makes the difference between the two fluxes decrease continuously till it becomes zero. Since rate of solidification is proportional to this flux imbalance, pull velocity decreases continuously and finally becomes zero.

Variation in concentration field in the melt with time for different crystal rotations has been shown in Figures 5.11 to 5.16. Concentration contours shown are for segregation coefficients, $k_0 = 0$ (Figures 5.11 to 5.13) and $k_0 = 0.2$ (Figures 5.14 to 5.16). No change in concentration of the melt takes place for $k_0 = 1$. Due to segregation concentration of the solute increases near the melt-crystal interface and this solute is then redistributed in the melt by convection currents. Concentration contours show how solute redistribution is caused by complex flow patterns in the melt. Rate of melt enrichment is maximum for $k_0 = 0$ as all the solute in the solidifying melt is rejected. Rejection for $k_0 = 0.2$[1] corresponds to the Nd:YAG case.

Figure 5.17 and 5.18 shows radial distribution of solute just below the melt-crystal interface. It can be seen that for no crystal rotation, radial variation in concentration is maximum with maximum concentration at the center of the crystal. This is simply a consequence of the purely buoyancy driven flow which makes the solute move towards crystal-center. Now as the crystal rotation is increased concentration profile starts becoming flat due to the centrifugal action at the melt-crystal interface.

[1] A value of $k_0 = 0.2$ has been found in the experiments of Lu [2000]

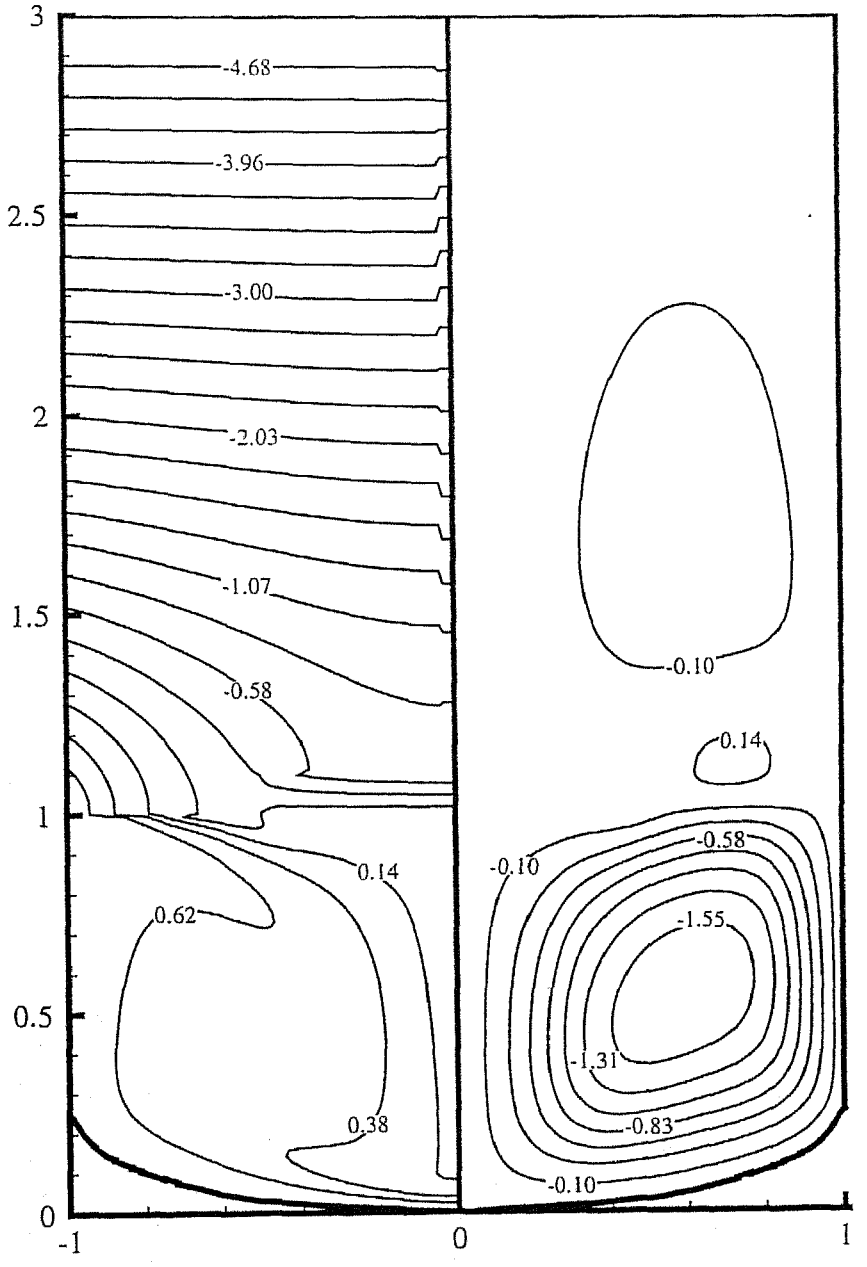


Figure 5.4: Temperature (isotherms on the left) and flow field (streamlines on the right) inside the Cz-system for Re = 0, Gr = 1×10^4 and melt height = 1.

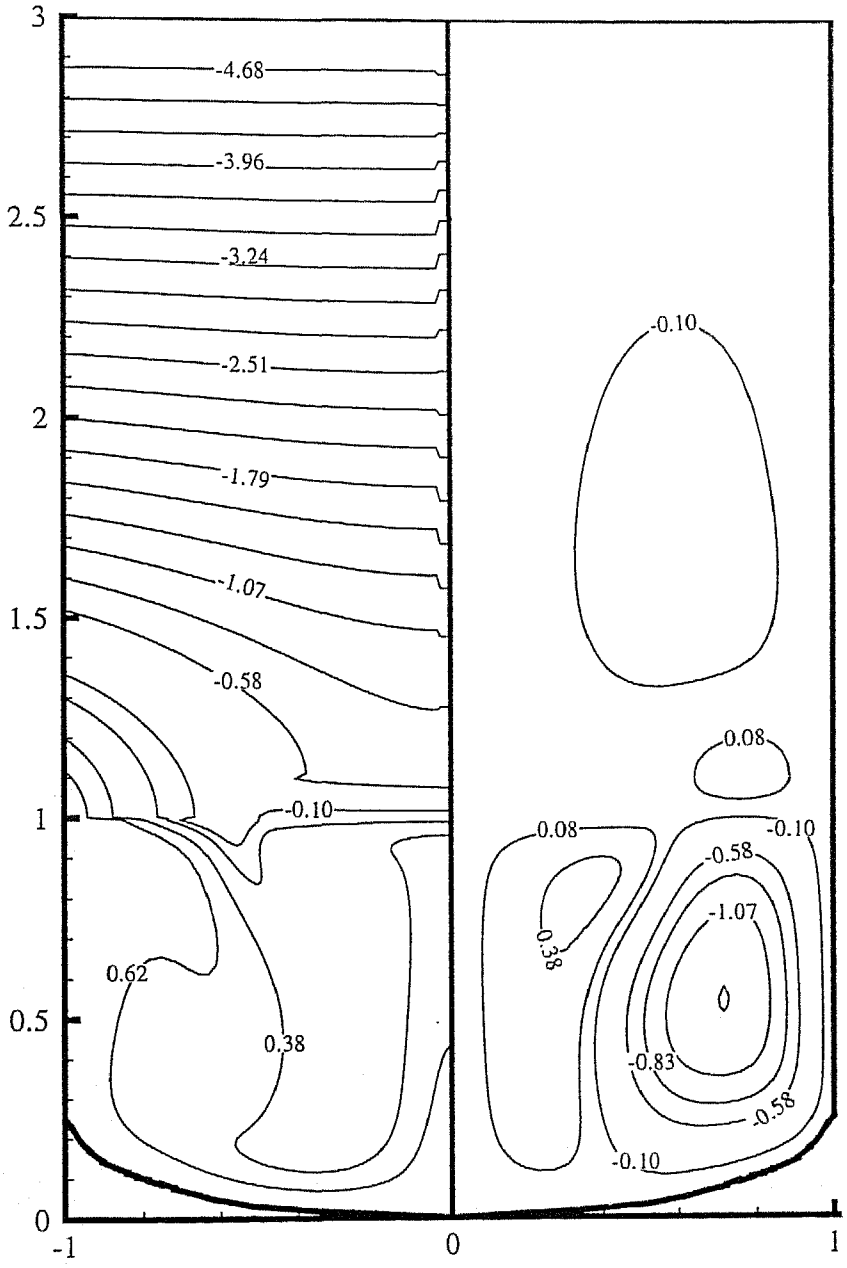


Figure 5.5: Temperature (isotherms on the left) and flow field (streamlines on the right) inside the Cz-system for Re = 250, Gr = 1×10^4 and melt height = 1.

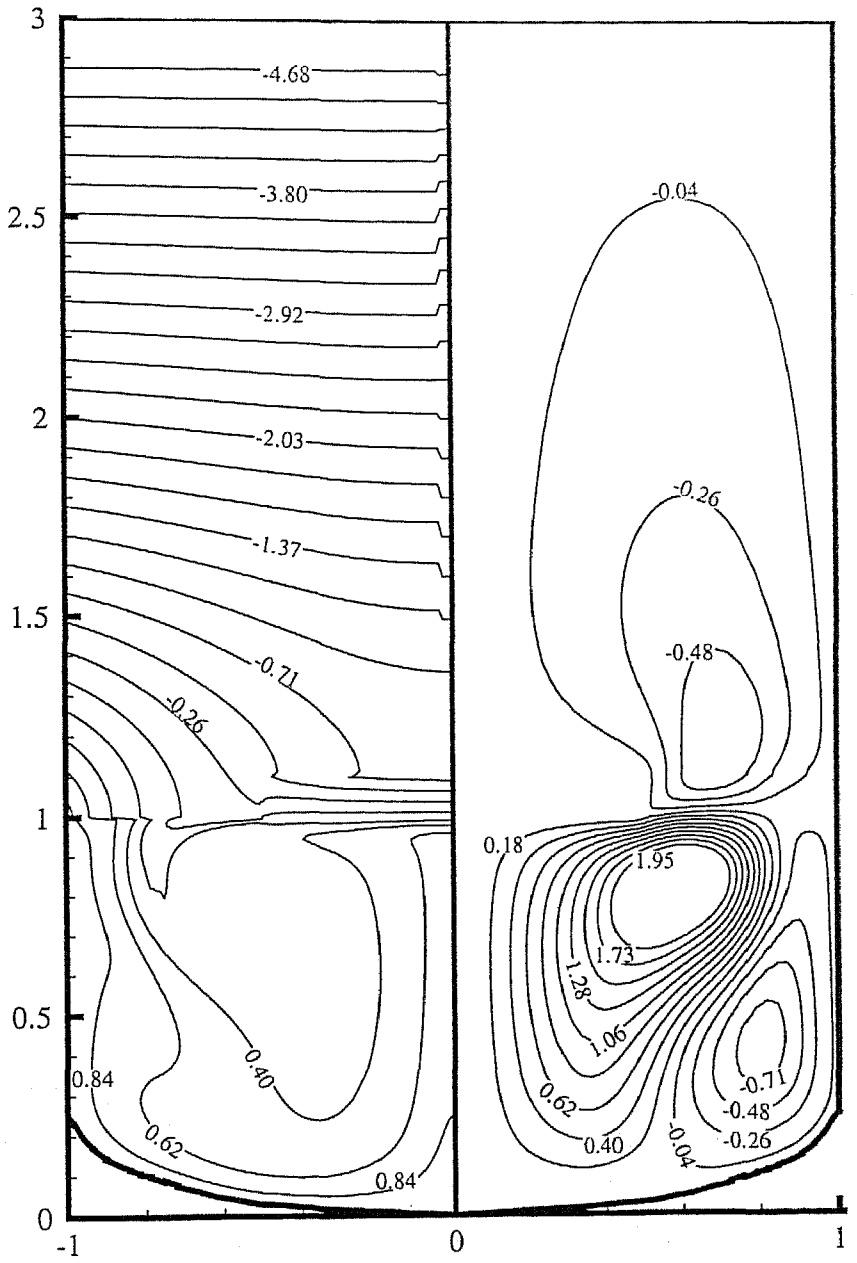


Figure 5.6: Temperature (isotherms on the left) and flow field (streamlines on the right) inside the Cz-system for Re = 500, Gr = 1×10^4 and melt height = 1.

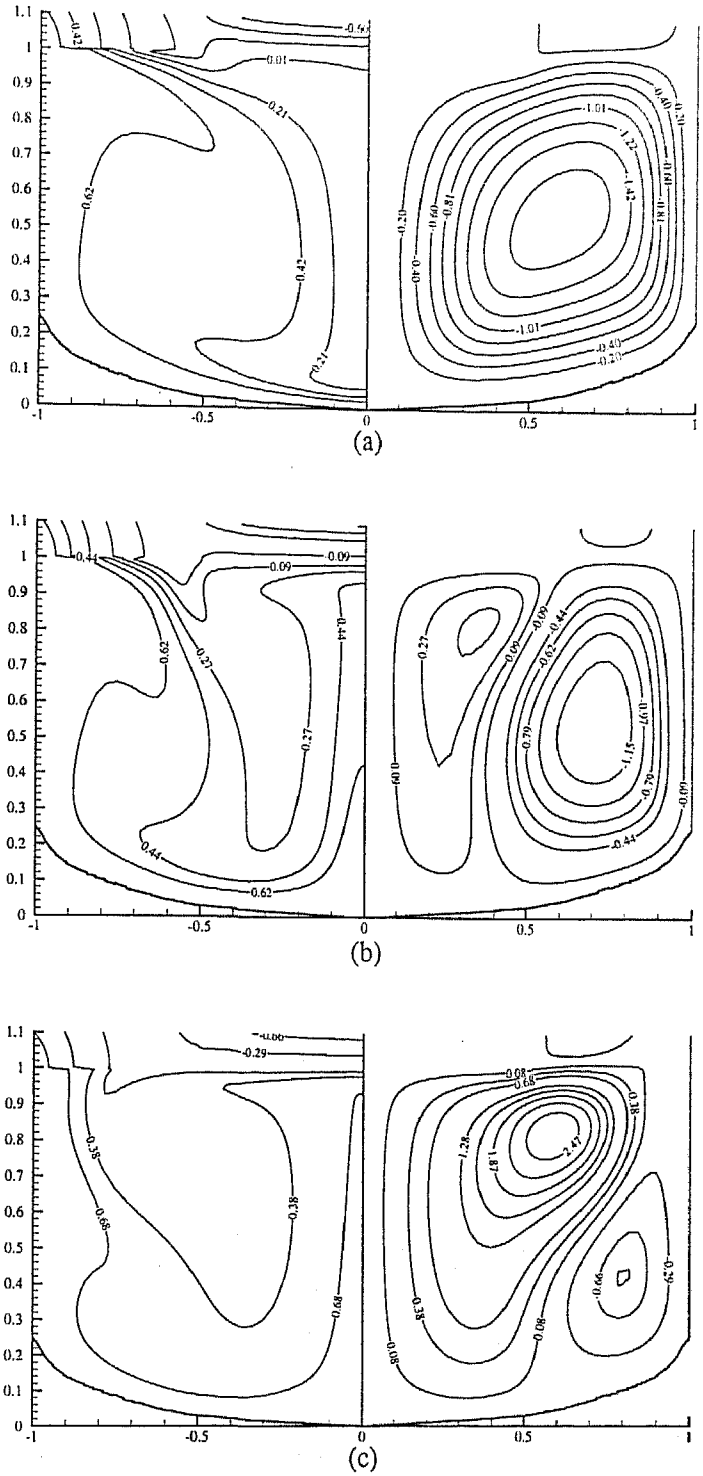


Figure 5.7: Temperature (isotherms on the left) and flow field (streamlines on the right) in Nd:YAG-melt for three different crystal rotations (*i.e.*, (a): Re = 0; (b): Re = 250 and (c): Re = 500). For all three crystal rotations, melt height and Grashof number are 1 and 1×10^4 respectively.

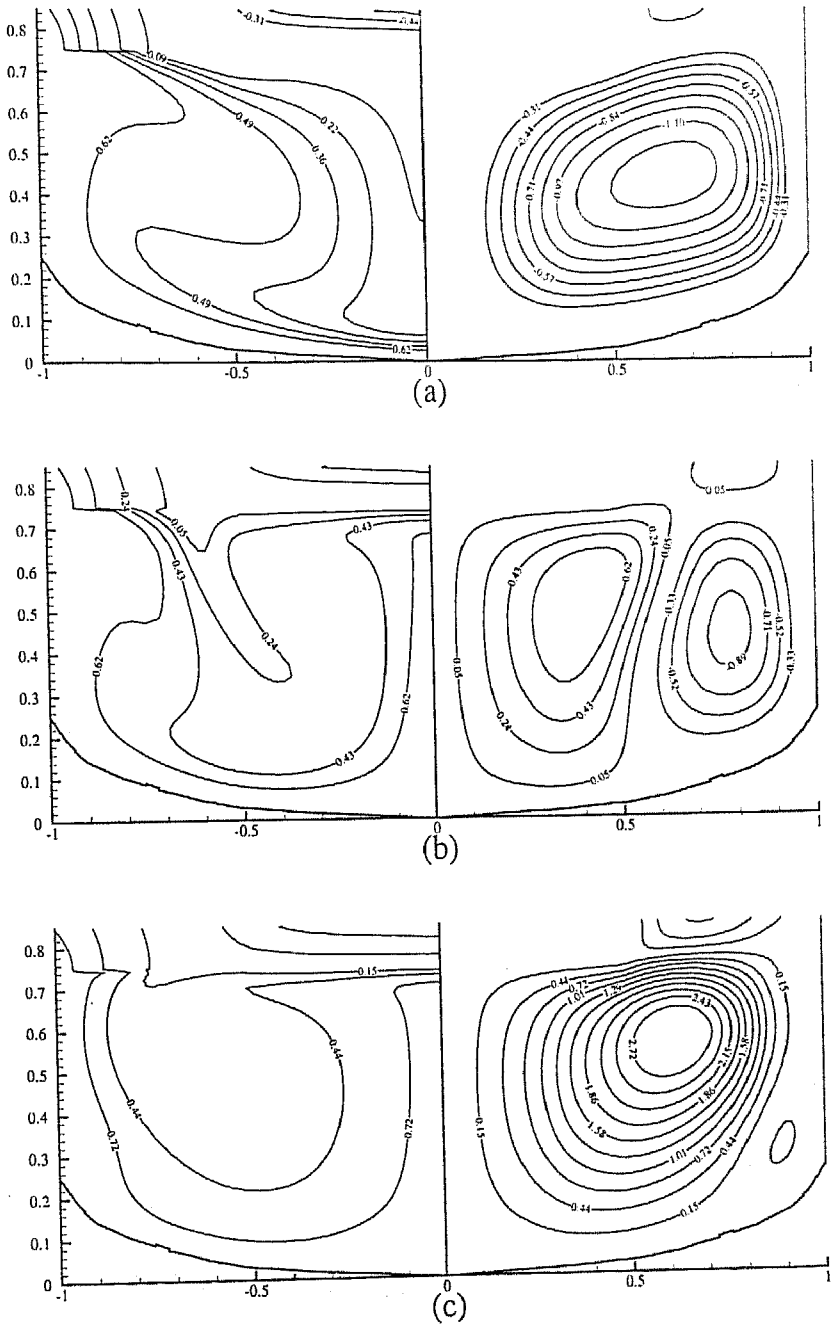


Figure 5.8: Temperature (isotherms on the left) and flow field (streamlines on the right) in Nd:YAG-melt for three different crystal rotations (*i.e.*, (a): Re = 0; (b): Re = 250 and (c): Re = 500). For all three crystal rotations, melt height and Grashof number are 0.75 and 1×10^4 respectively.

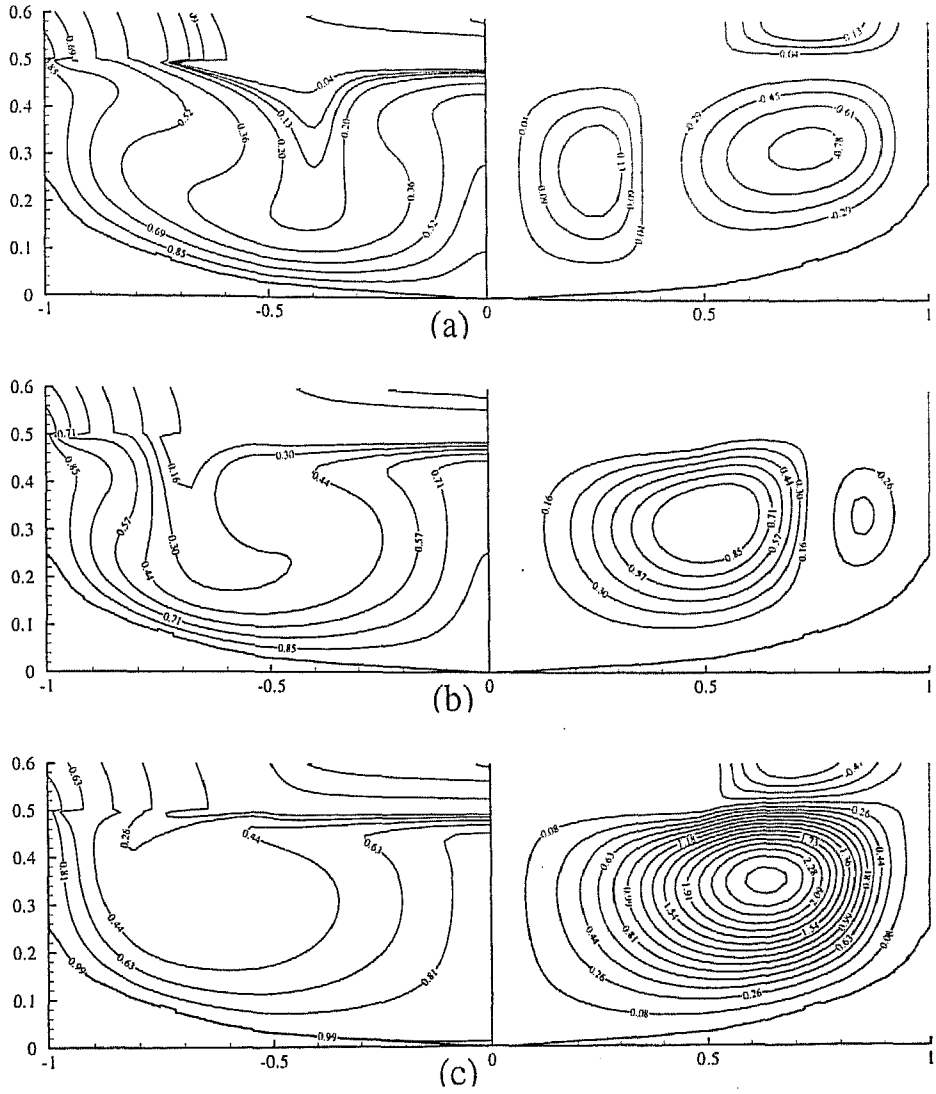


Figure 5.9: Temperature (isotherms on the left) and flow field (streamlines on the right) in Nd:YAG-melt for three different crystal rotations (*i.e.*, (a): Re = 0; (b): Re = 250 and (c): Re = 500). For all three crystal rotations, melt height and Grashof number are 0.5 and 1×10^4 respectively.

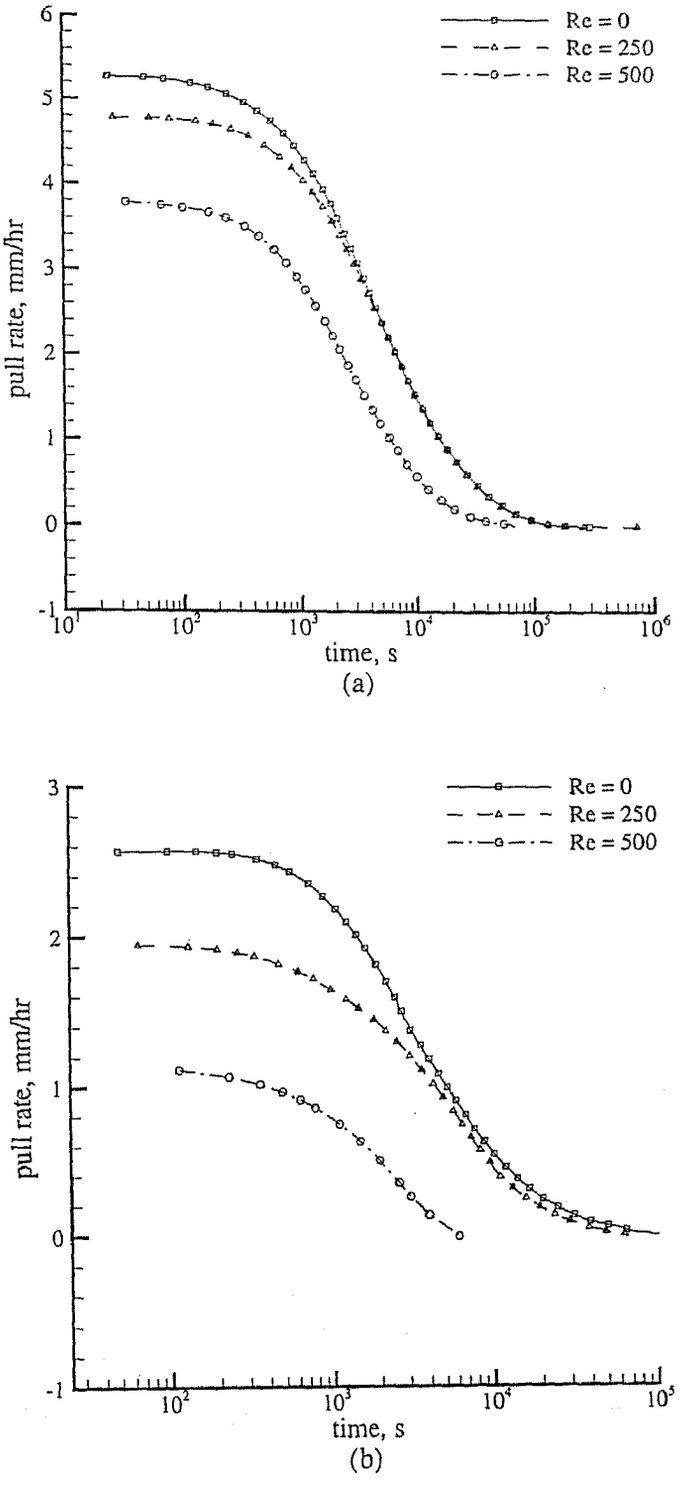


Figure 5.10: Variation in pull velocity of a continuously growing Nd:YAG crystal for two different initial melt heights, *i.e.*, (a): 1 and (b): 0.5. In both the cases, initial crystal height is 0.1 and Gr $= 1 \times 10^4$. Re is Reynolds number for the crystal rotation.

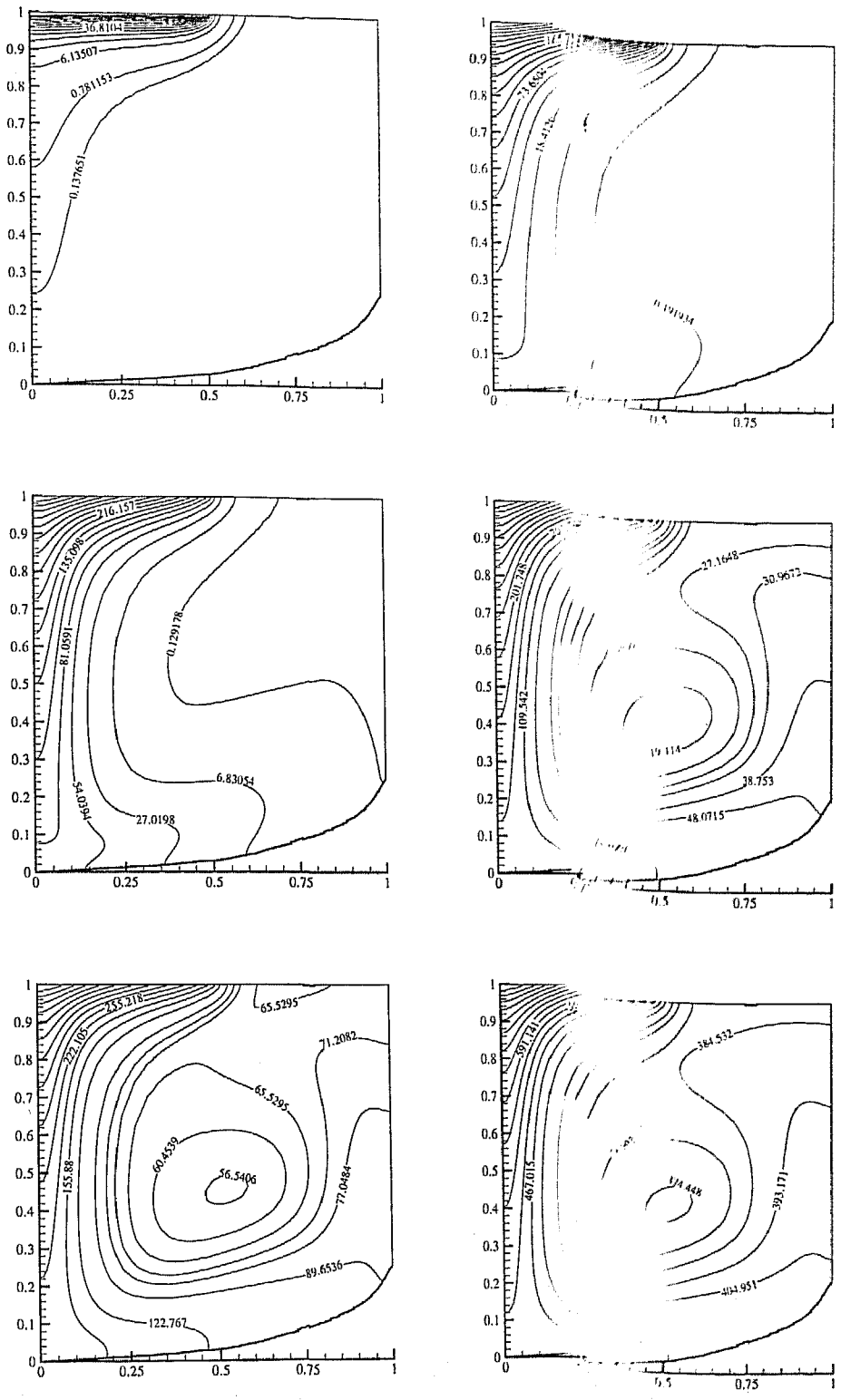


Figure 5.11: Iso-concentration lines for Nd concentration in the melt at six different values of time, *i.e.*, (a) 0.01, (b) 0.05, (c) 0.1, (d) 0.5, (e) 1 and (f) 5. Contour variable is taken as $(C - C_0) \times 10^5$, where $C_0(= 1)$ is the initial solute concentration. Other parameters are Gr $= 1 \times 10^4$, Re $= 0$, segregation coefficient $k_0 = 0$ and pull rate $= 3$ mm/h.

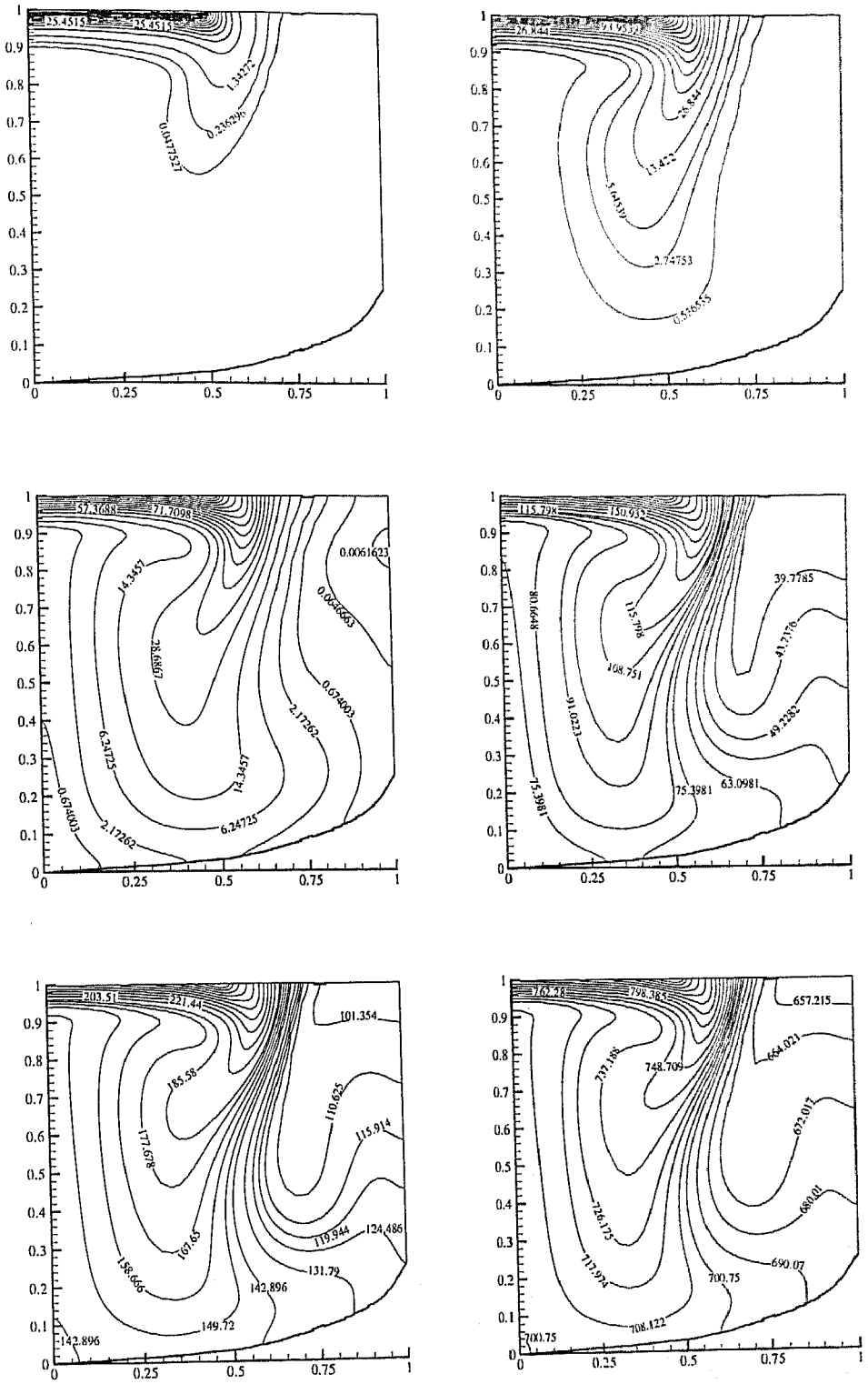


Figure 5.12: Iso-concentration lines for Nd concentration in the melt at six different values of time, *i.e.*, (a) 0.01, (b) 0.05, (c) 0.1, (d) 0.5, (e) 1 and (f) 5. Contour variable is taken as $(C - C_0) \times 10^5$, where $C_0(= 1)$ is initial solute concentration. Other parameters are $\mathrm{Gr} = 1 \times 10^4$, $\mathrm{Re} = 250$, segregation coefficient $k_0 = 0$ and pull rate $= 3$ mm/h.

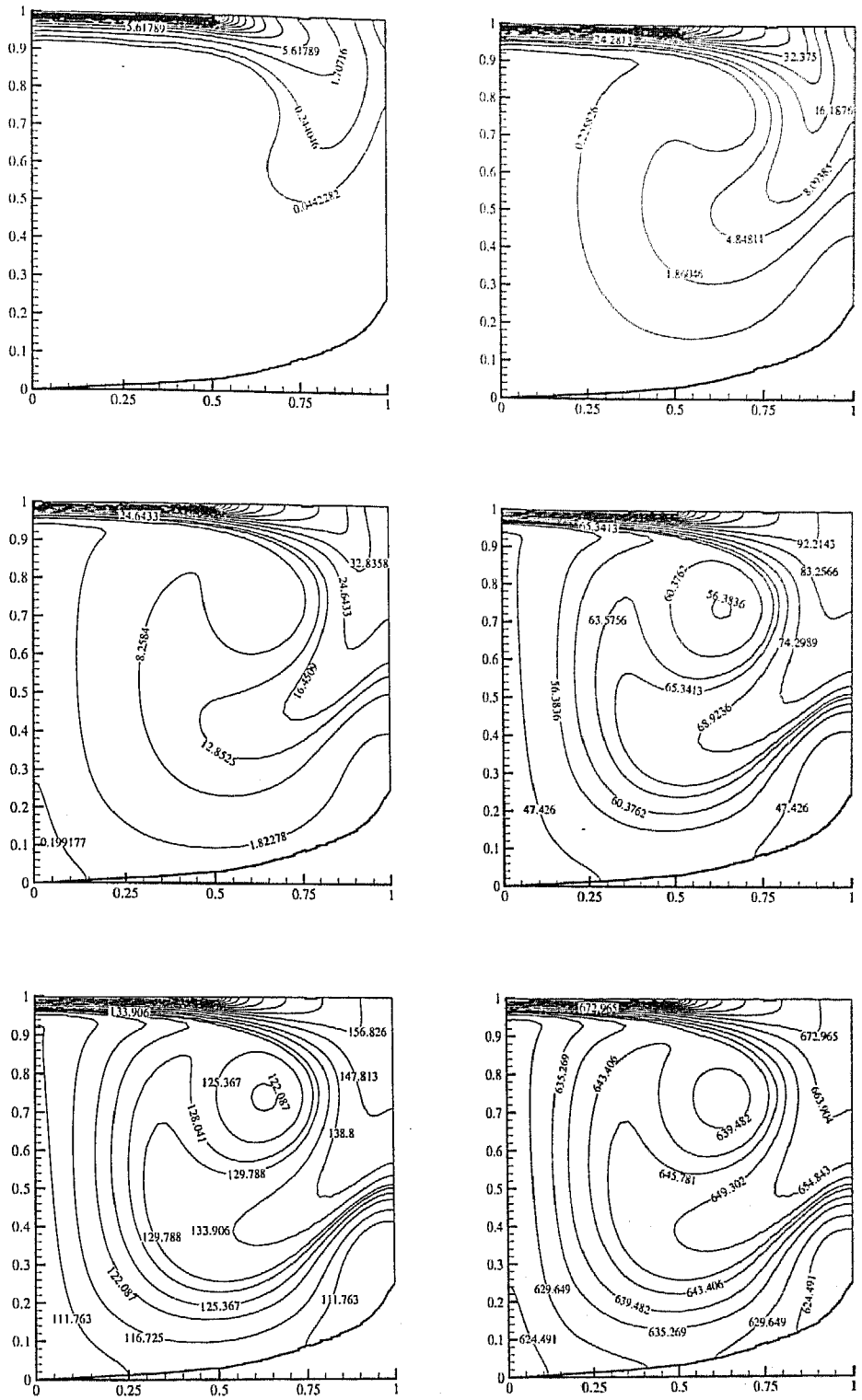


Figure 5.13: Iso-concentration lines for Nd concentration in the melt at six different values of time, *i.e.*, (a) 0.01, (b) 0.05, (c) 0.1, (d) 0.5, (e) 1 and (f) 5. Contour variable is taken as $(C - C_0) \times 10^5$, where $C_0(= 1)$ is initial solute concentration. Other parameters are Gr $= 1 \times 10^4$, Re $= 500$, segregation coefficient $k_0 = 0$ and pull rate $= 3$ mm/h.

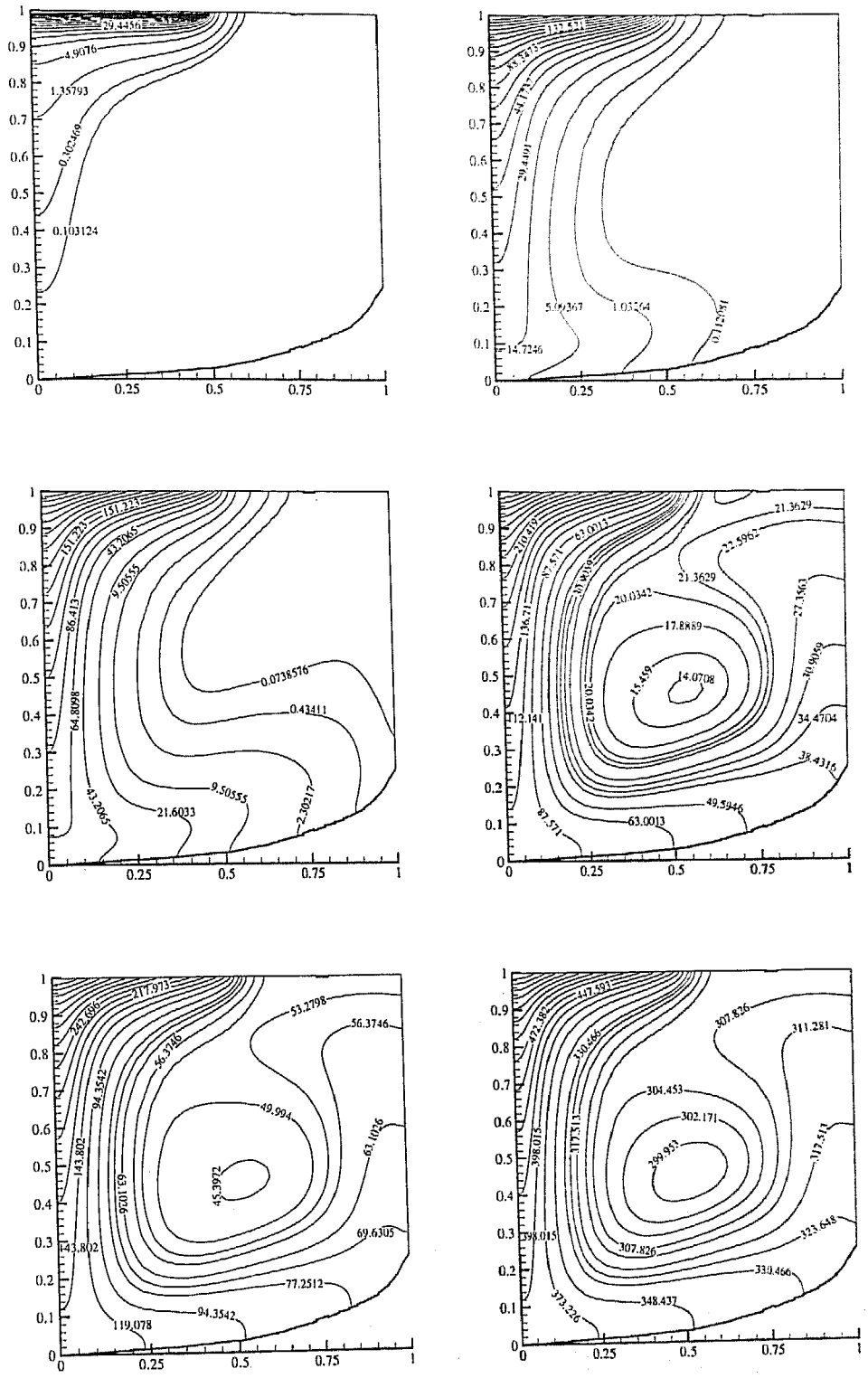


Figure 5.14: Iso-concentration lines for Nd concentration in the melt at six different values of time, *i.e.*, (a) 0.01, (b) 0.05, (c) 0.1, (d) 0.5, (e) 1 and (f) 5. Contour variable is taken as $(C - C_0) \times 10^5$, where $C_0(= 1)$ is initial solute concentration. Other parameters are Gr $= 1 \times 10^4$, Re $= 0$, segregation coefficient $k_0 = 0.2$ and pull rate $= 3$ mm/h.

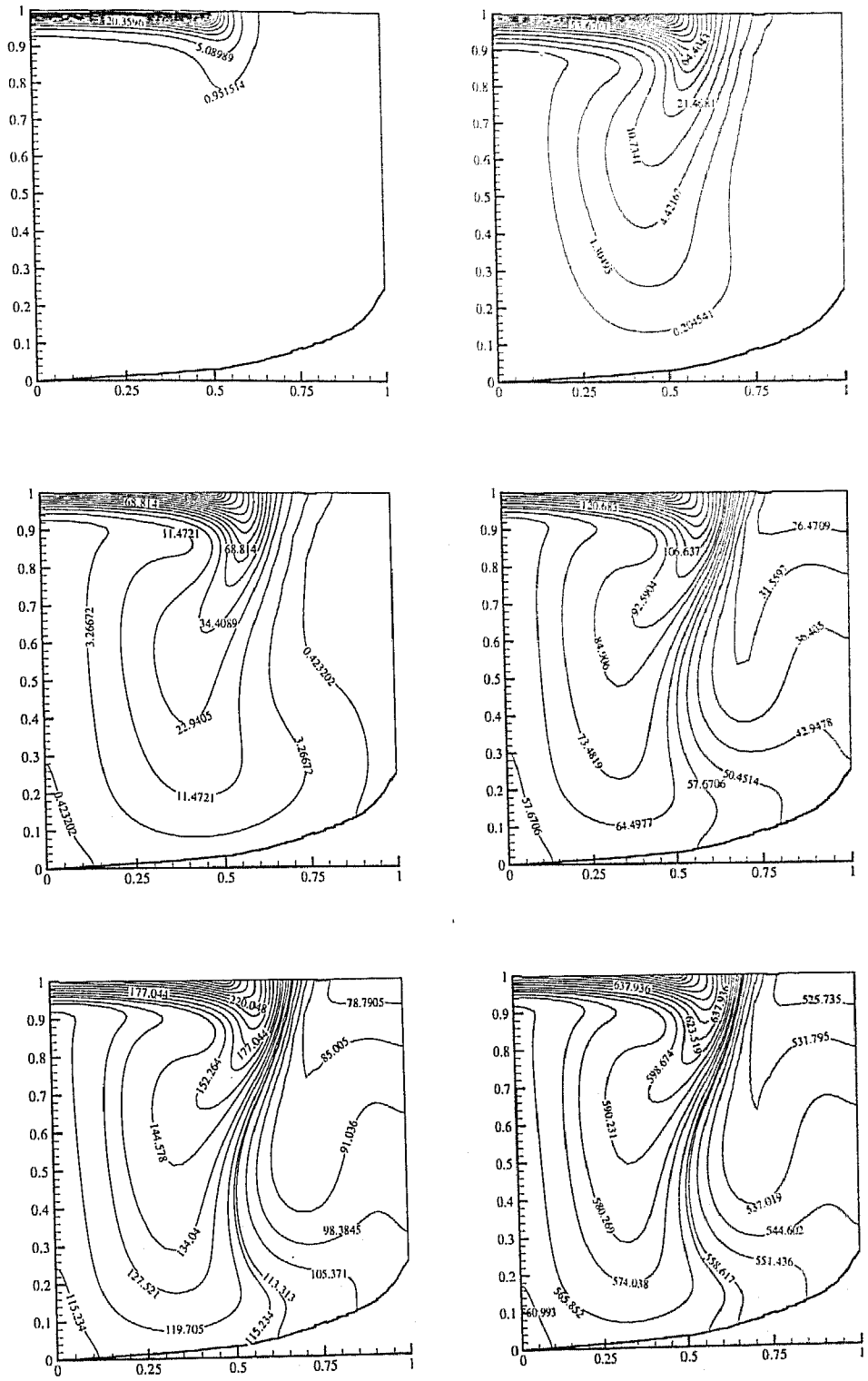


Figure 5.15: Iso-concentration lines for Nd concentration in the melt at six different values of time, *i.e.*, (a) 0.01, (b) 0.05, (c) 0.1, (d) 0.5, (e) 1 and (f) 5. Contour variable is taken as $(C - C_0) \times 10^5$, where $C_0(= 1)$ is initial solute concentration. Other parameters are Gr $= 1 \times 10^4$, Re $= 250$, segregation coefficient $k_0 = 0.2$ and pull rate $= 3$ mm/h.

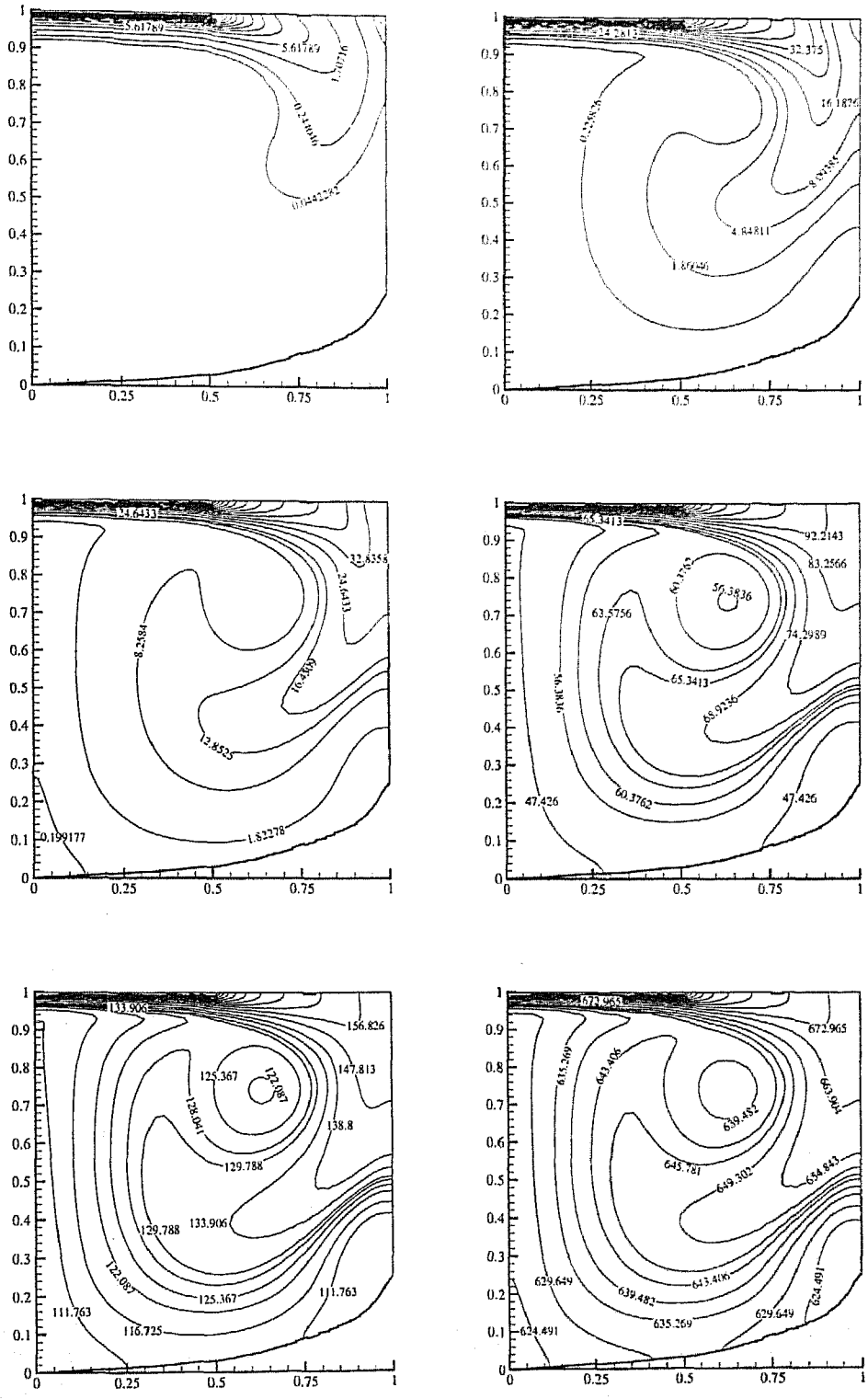


Figure 5.16: Iso-concentration lines for Nd concentration in the melt at six different values of time, *i.e.*, (a) 0.01, (b) 0.05, (c) 0.1, (d) 0.5, (e) 1 and (f) 5. Contour variable is taken as $(C - C_0) \times 10^5$, where $C_0(= 1)$ is initial solute concentration. Other parameters are Gr $= 1 \times 10^4$, Re $= 500$, segregation coefficient $k_0 = 0.2$ and pull rate $= 3$ mm/h.

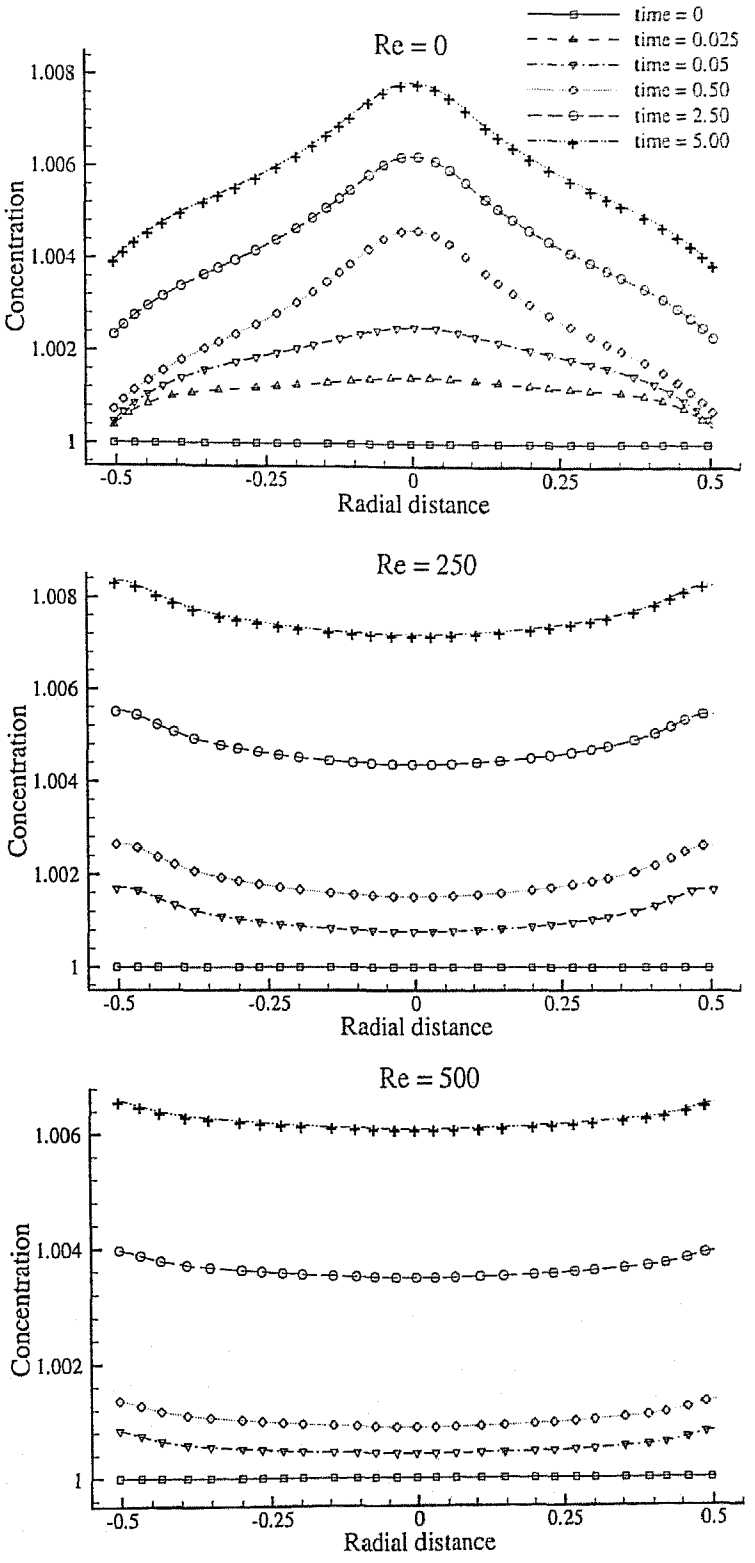


Figure 5.17: Radial concentration profiles at different times for three different values of crystal rotations. Other parameters are, initial melt height = 1, Gr = 1×10^4, segregation coefficient $k_0 = 0$ and pull rate = 3 mm/h.

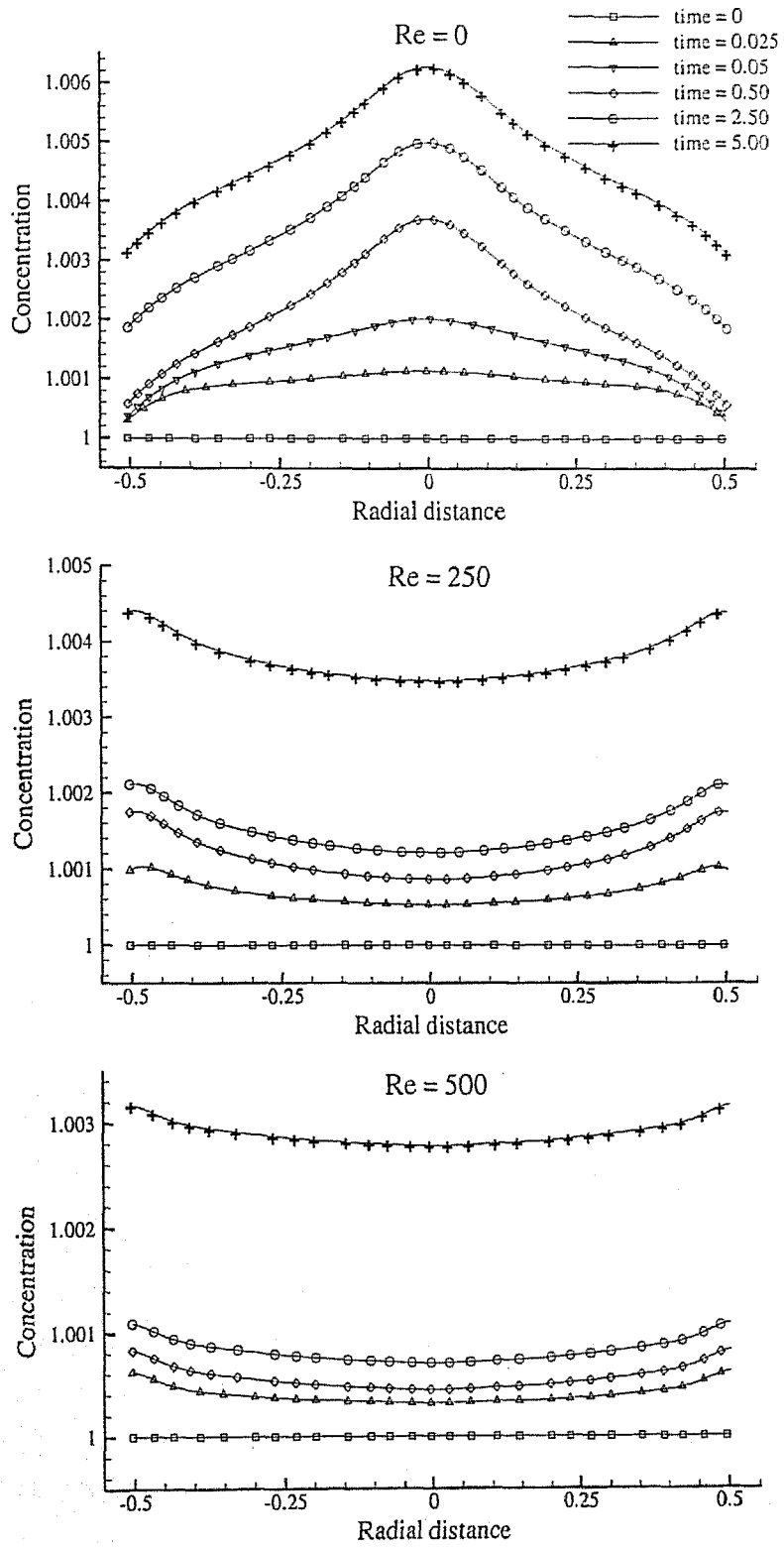


Figure 5.18: Radial concentration profiles at different times for three different values of crystal rotations. Other parameters are, initial melt height = 1, Gr = 1×10^4, segregation coefficient $k_0 = 0.2$ and pull rate = 3 mm/h.

Chapter 6

Conclusions and Scope for Future Work

6.1 Conclusion

The Preconditioned Conjugate Gradient (PCG) algorithm has been tested for various benchmark problems in heat transfer. Variation in growth rate, melt convection and solutal transport have been studied for the growth of Nd:YAG single crystals by the Czochralski process. Following major conclusions have been arrived at in the present work.

1. CPU times for the test case of unsteady-nonlinear heat conduction show that PCG is extremely effective for matrix inversion as compared to Gaussian elimination and the Gauss-Seidel technique. Moreover, PCG becomes an even more attractive choice on finer grids.

2. PCG is well-suited for matrix inversion in advection-diffusion problems. The results obtained for Nusselt number in channel flow match the analytical solution.

3. Eigenvalue spectrums of the original and preconditioned matrices show that preconditioning with incomplete lower-upper (ILU) decomposition is very effective. Moreover, it is general in the sense that it can be constructed and applied for any nonsingular coefficient matrix.

4. In the present simulation for crystal growth, crystal pull rate is varied to grow constant diameter crystals. The results obtained for pull-velocity variation show that it is not possible to maintain favorable growth conditions

for a long period of time. To grow longer constant diameter crystals, parameters such as crucible temperature and enclosure temperature need to be varied.

5. Redistribution of the segregated solute in the melt takes place because of the complex flow patterns in the melt. Moreover, it is not possible to grow crystals with uniform radial solute distribution without any crystal rotation.

6.2 Scope for Future Work

The present work has opened up new areas of exploration. Following challenges are recommended for future researchers.

1. An extensive assessment of various preconditioning techniques for PCG is required.

2. Attractive parallel properties of the effective PCG algorithm calls for the development of appropriate algorithms.

3. An extensive study of the effect of various parameters on crystal growth is needed to provide guidelines for growing high quality single crystals.

4. Exact prediction of solute incorporation into the grown crystals needs a microscopic model for solute segregation.

[10] Golub, G. H. and O'Leary, D. P., Some history of the conjugate gradient and Lanczos methods, *SIAM Review*, Vol. 31, pp. 50, 1989.

[11] Golub, G. H. and Ortega, J. M., *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, San Diego, 1993.

[12] Hestenes, M. R. and Stiefel, E. L., Methods of conjugate gradients for solving linear systems, *Journal of Research of National Bureau of Standards*, Vol. 49, pp. 409, 1952.

[13] Kershaw, D. S., The incomplete Cholesky conjugate gradient method for iterative solution of system of linear equations, *Journal of Computational Physics*, Vol. 26, pp. 43, 1978.

[14] Kobayashi, N., Computational simulation of the melt flow during Czochralski growth, *Journal of Crystal Growth*, Vol. 43, pp. 357-363, 1978.

[15] Kopetsch, H., Numerical simulation of the Czochralski bulk flow of Silicon on a domain confined by a moving crystal-melt interface and a curved melt-gas meniscus , *Physicochemical Hydrodynamics*, Vol. 11, No. 3, pp. 357-375, 1989.

[16] Lanczos, C., Solution of system of linear equations by minimized iterations, *Journal of Research of National Bureau of Standards*, Vol. 49, pp. 33, 1952.

[17] Leonard B. P., A Stable and Accurate Convective Modeling Procedure based on Quadratic Upstream Interpolation, *Comput. Methods Appl. Mech. Eng.*, Vol. 19, pp. 59-98, 1979.

[18] Lu, J., Prabhu, M., Song, J., Li, C., Xu, J., Ueda, K., Kaminskii, A. A., Yagi, H. and Yanagitani, T., Optical properties and highly efficient laser oscillation of Nd:YAG ceramics, *Applied Physics*, Vol. B 71, pp. 469-473, 2000.

[19] Meijerink, J. A. and Vorst, H. A. van der, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, *Mathematics of Computations*, Vol. 31, pp. 148, 1977.

[20] Meijerink, J. A. and Vorst, H. A. van der, Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems, *Journal of Computational Physics*, Vol. 44, pp. 134-155, 1981.

[21] Minkowycz, W. J., Sparrow, B. M., Scheider, G. E., Pletcher, R. H., *Handbook of Numerical Heat Transfer*, J. Wiley & Sons Inc., U.S.A., 1988.

[22] Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere, Washington, DC, 1980.

[23] Pissanetzky, S., *Sparse Matrix Technology*, Academic Press, New York, 1984.

[24] Prasad, V., Zhang, H. and Anselmo, A.P., Transport phenomena in Czochralski crystal growth processes, *Advances in Heat Transfer*, Vol. 30, pp. 313-435, 1997.

[25] Reid, J. K., On the method of conjugate gradients for the solution of large sparse systems of linear equations, *Large Sparse Sets of Linear Equations*, edited by J. K. Reid, pp. 231, Academic Press, New York, 1971.

[26] Saad, Y., *Iterative Methods for Sparse Linear Systems*, PWS Publishing Co., Boston, 1996.

[27] Sheorey, T., Modeling of Enhanced Oil Recovery from a Porous Formation, Doctoral dissertation, Indian Institute of Technology Kanpur, Kanpur, 2001.

[28] Sheorey, T., Muralidhar, K. and Mukherjee, P. P., Numerical Experiments in the Simulation of Enhanced Oil Recovery, *International Journal of Thermal Sciences*, Vol. 40(11), pp. 981-997, 2001.

[29] Sheorey, T., and Muralidhar, K., Application of Domain Decomposition to the Simulation of Oil Recovery on a Parallel Computer, *Institution of Engineers Journal*, Chemical Engineering Division, Vol. 82, pp. 24-29, 2001.

[30] Sheorey, T., Muralidhar, K. and Mukherjee, P. P., Isothermal and Non-isothermal Oil-water Flow and Viscous Fingering in a Porous Medium, *International Journal of Thermal Sciences*, Vol. 42(7), pp. 665-676, 2003.

[31] Tannehill, J. C., Anderson, D. A. and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Second Edition, Taylor & Francis, Washington, 1997.

[32] Thomsan, J. F., Warsi, Z. U. A. and Martin, C. W., Numerical Grid generation-Foundation and applications, *North Holland publishers*, 1985.

[33] Turkel, E., Preconditioning Techniques in Computational Fluid Dynamics, *Annual Review of Fluid Mechanics*, Vol. 31, pp. 385-416, 1999.

[34] Van de Velde, E. F., *Concurrent Scientific Computing*, Springer-Verlag, New York, 1994.

[35] Watkins, D. S., *Fundamentals of Matrix Computations*, John Wiley & sons, Inc., New York, 2002.

[36] Zhang, H. and Prasad, V., A multizone adaptive process model for low and high pressure crystal growth, *Journal of Crystal Growth*, Vol. 155, pp. 47-65, 1995.

[37] Zhang, H, Prasad. V., and Moallemi, M.K., Numerical Algorithm using Multizone Adaptive Grid Generation for multiphase transport processes with moving and free boundaries, *Numerical Heat Transfer*, Part B, Vol. 29, pp. 399-421, 1996.

[38] Zou, Y. F., Wang, G. X., Zhang, H., Prasad. V. and Bliss, D.F., Macro-segregation, dynamics of interface and stresses in high pressure LEC grown crystals, *Journal of Crystal Growth*, Vol. 180, pp. 524-533, 1997.